Linux Journal Issue #74/June 2000

When the expense of a remote access server is no longer attractive, it's time to look at the solution offered by a VPN.

In this episode, Mr. Hammel tells us about the Window Maker window manager, a less flashy but more mature product than Enlightenment.

Reviews

Columns

Scanning photos has never been easier.

This company converted to a Linux server for its Oracle database and increased operation speeds eight-fold.

In this article we are going to cover the idea of logging the system activities over a TCP network in a secure way, by interfacing the existing syslog daemon with secure shell using simple Perl scripts.

To make changes in the kernel, you need to know all about the diff and patch commands.

Building a web-based telephone book can be easy, as long as you don't drink too much of Marcel's wine while you work.

Think Linux is only for the PC? Think again.

Focus: People Behind Linux

Marjorie Richardson

Issue #74, June 2000

All in all, we've had a good time with this one—you will too.

Wow!, what a big topic this turned out to be—so many people, so little space. After making what seemed like an infinite list of people we wanted to talk to and honor, we decided we better figure out a way to narrow our focus. We did this by using the Kernel 1.0 credits file—these people are truly the “founding fathers”--and adding a few more names, including Patrick Volkerding and Alessandro Rubini. Even so, we still had 84 names. Knowing we might not be able to find everyone (how right we were!), we dug in and started sending out e-mails, asking for phone numbers and interviews.

All those we were able to contact were gracious and cooperative, sending us e-mail copies of their answers so that we can include them on the web site in coming weeks. Having just talked to Linus in September of last year, I decided to write a short bit about him without actually contacting him again. We'll save that for the next kernel release. I think you're going to enjoy learning a bit about the people who brought us our favorite operating system.

If Linus is the “father” of Linux (and we all agree he is), then “maddog” is the “godfather”. He is the glue that holds the Linux community together, and he shares all with us this month in an article and a centerfold.

All in all, we've had a good time with this one—you will too.

A Visit with Caldera

Some folks from Caldera came by the first week of April to tell us what's new in Utah. After their change in focus from “Linux for Business” to “Linux for eBusiness”, Caldera has gone solidly after the e-commerce market, with three new product releases: eDesktop, eServer and eBuilder. All three products have been optimized for use on the Internet. eDesktop 2.4 is the latest release of the

Caldera OpenLinux distribution with enhancements (improved hardware detection by Lizard) and additions, such as the Citric ICA client which provides access to Windows applications through the Web, and MoneyDance, a personal checkbook-type application comparable to Quicken. Also, remote administration can be done through the Web as well as unattended installs across main machines. eServer provides Pentium-class operation and is free. eBuilder is the big one, a combination of Open Linux eServer, Evergreen's Ecential 3.0 and IBM's WebSphere. It is designed to give the e-commerce site everything it needs, e.g., merchandise and multimedia database, search engine, order distribution, shopping cart and payment processing.

The eBuilder product is directed toward the big business customer who wants to get into the e-commerce market in a hurry. It is modular, distributed and easy to customize and manage. It is also *very* expensive—many thousands of dollars. At Comdex 2000, Ransom Love said Linux is a “proprietary” platform, and with eBuilder, it becomes a very commercial one for Caldera.

—Marjorie Richardson, Editor in Chief

Contents

Archive Index Issue Table of Contents

Advanced search

We Talk to Everybody

Marjorie Richardson, Jason Schumaker and David Penn

Issue #74, June 2000

A quick look at some of the people who helped make Linux possible.

The editors here at *Linux Journal* have been talking about doing a Who's Who in Linux for some time, but couldn't decide on a way to do it that would make sense and not need the space of a book. With so many people contributing to Linux and new ones joining the community every day, it would be almost impossible to compile a list that didn't leave someone out. We finally decided that what we wanted to do was present short profiles of the developers who contributed to kernel 1.0, and of a few others who were involved at the time of its release. We got the kernel 1.0 credits file and split it up. I took a few names, including Linus', having talked to him just last year. With the help of Jason Kroll, we spent a good bit of time tracking people down, and were not entirely successful. A list of those whom we couldn't get hold of and the contribution they made to the kernel is included as a sidebar. If any of you guys are reading this, please get in touch (info@linuxjournal.com.com)—we'd like to talk to you.

We wish to thank all these people for talking to us and providing this look at who they were then and now. Since there was not room to include each interview in its entirety, we will put the interviews on the *LJ* web site in the coming weeks. We also thank these folks for their invaluable contributions to Linux—without them, where would it be today? Or *LJ*, for that matter?

Linus Torvalds

Everyone knows who Linus Torvalds is. He's the metaphoric father of the Linux community, having brought the Linux operating system into being. After doing so, he kicked it out of its nest and onto the Internet and invited others to try it out and participate in its growth, not realizing at the time what a tiger he had let loose on the world. He is a quiet, self-assured young man, a family man who obviously loves his daughters, allowing them to join him onstage at conferences they attend with him and his wife Tove.

Writing Linux earned Linus his master's degree at Helsinki University. Since then, he has turned kernel maintenance over to Alan Cox. Development is done by many others, yet Linus remains the mainstay of Linux. He makes the final decisions about what will and will not go into the kernel and when a new version will be released. His word is law, and no one disputes it.

After leaving the university, Linus moved to the United States with Tove, near San Jose, California, and went to work for the ultra-secret Transmeta. Early this year, Transmeta announced their Crusoe chip, which is headed for the mobile appliance market. Linus had been helping by making a Linux version that would fit in flash ROM—here's what he said about it in a conversation with Doc Searls.

> It's the standard 2.3 kernel with some power management stuff done on it, but anybody can see what that is. There is the compressed file system that I made part of the standard kernel not long ago. And that's really the question of... you have a very limited amount of ROM. You want to fit Netscape in there, too. And you want to page things in from the ROM. You obviously want to compress it. That's number one. But how do you compress it so you can still do random seeks and paging in things? Those are the kinds of questions I spent a fair amount of time thinking about. What's the good way of doing this? What fits in and works well? These are not fundamental design shifts.

Linus agrees to make appearances at many trade shows, although these days, he prefers question-and-answer sessions over giving talks. In fact, he agrees to so many that I once wondered if he knew how to say “No”. (He does.) But he also realizes he is a big draw, and helping Linux and Linux shows become

successful is something he is willing to do. He remains unassuming, and says fame has not intruded on his personal life.

Linus has proven to be a good father to both his children and Linux. The community couldn't have a better leader.

Donald Becker

Donald didn't respond to any of our messages, but we know a bit about him so we decided to wing it. First of all, we know Donald because he was on the cover of issue 10, February 1995, where he was a part of the Linux Conference at Open Systems World in Washington, DC.

Donald was a big contributor early in the Linux game, writing the code for the Ethernet drivers. He did this while working for NASA at the Center of Excellence in Space Data and Information Sciences as a staff scientist. Information on Donald and his code can be found at cesdis.gsfc.nasa.gov/people/becker/whoiam.html.

He was the principal investigator on NASA's Beowulf Project, “an effort to develop a software distribution to help others build high-performance workstations based on a cluster of off-the-shelf processing nodes running Linux.”

He is now the Chief Technical Officer of Sycld Computing Corporation, where he “continues the Beowulf work, making available the expertise to non-NASA entities.”

For fun, Donald likes to kayak, and you can read about his kayaking trip down the Yukon River in 1993 at his web site. There is also a beautiful picture of the river and mountains, as well as information on how to reach him.

Mark Bolzern

Mark discovered Linux back in 1992, and by 1994 was predicting it would be the next-generation UNIX and challenge even Microsoft. That's what I call prescient. Rather than become a Linux developer, he instead took the road of an advocate.

When he started his business, Workgroup Solutions, he sold one of the first commercial products for Linux: Multisoft's FlagShip. In fact, he persuaded Multisoft to port it to Linux in the first place. He also sold his own distribution, Linux Pro, based first on Slackware and later on Red Hat. It was always a bit behind the cutting edge, but this meant it was always stable, a definite plus for the business market at which he was aiming. Today, with all the advances in Linux distributions, he no longer feels there is a need for Linux Pro and has withdrawn it from the market. Workgroup Solutions has turned into LinuxMall, and as a business proves that supporting Linux can bring success.

When asked what he liked most about Linux, Mark replied, “Its openness, its use of the GPL, its ability to educate those willing to take responsibility to educate themselves, and finally, its flexibility to adapt to the needs of anyone educationally empowered or able to hire someone who is.” He strongly compares Linux and American history in this way: “I see the Linux movement as a parallel to the American Revolution, where the GPL is analogous to the Constitution. This, I guess, makes Richard Stallman a Thomas Jefferson, and perhaps `The Cathedral and the Bazaar' is the Declaration of Independence.”

Mark dedicates his life to his business and Linux advocacy, finding no time for a family or even a vacation. However, he doesn't feel he is wasting his life on computers, but rather providing a service that will benefit others and perhaps even the whole world, when Linux achieves “world domination”. His belief in serving others is proven through the way he runs his company. He hires people who are already a part of the Linux community, and gives a percentage of the company's gross profit back to the community in support of Linux events, projects and public relations work.

An all-around nice guy, Mark is a familiar face at the trade shows. His web page is http://www.linuxmall.com/, and he can be reached at mark@linuxmall.com.

Andries Brouwer

When it comes to Linux, mathematician Andries Brouwer has a simple explanation as to why it worked for him:

> I tried it and it worked. In fact, it worked really well. FreeBSD I tried later, and I preferred Linux.

Unlike many of those whose hacking of the Linux kernel was a natural by-product of their work as a computer scientist, programmer or software developer, Andries, who managed international character handling for the Linux keyboard and console, arrived at Linux through a different route.

> I use my machines mostly for mathematics—(for) both computations and writing papers and books. Whenever something is wrong, I like to fix it, and this brought me to various parts of the kernel, mostly with small fixes, sometimes with slightly larger code fragments.

But working on Linux over the Internet, in fact working over the Internet itself, is nothing new to Andries, who today still considers himself a mathematician.

> I released Hack on the Net, maybe around 1984. This led to lots of contacts with many people all over the Net, both via e-mail and via net.games.hack. So, to me,

> cooperation over the Net was a well-known concept, not revolutionary at all.
>
> Andries Brouwer's e-mail address is aeb@cwi.nl.

Ed Carp

Ed Carp, whose contributions to the Linux kernel include work on cron, UUCP, Elm, Pine and pico, first ran into Linux in late 1991 when someone forwarded him the Linux project announcement from a BSD newsgroup.

> I was attracted to Linux because of the distributed development model. I had looked at 386BSD, but the development model was completely controlled by William Jolitz, who made releases about every six months, and I couldn't wait for bug fixes that long.

Sound familiar? At the time, Ed was working at Sun Microsystems, “downloading a full newsfeed via a Telebit modem to a XENIX box.” When his upstream newsfeed switched to HDB UUCP, and the XENIX UUCP could not support the new protocols, Ed went searching. “When I called SCO, they quoted me $1500 to upgrade my very old XENIX. So I was in the market for something cheaper and that I could tweak the source if I needed to.”

Insofar as Ed's Linux hacking rose directly from his computer needs, UUCP, Sendmail and Elm were among his first Linux contributions. Of the times, Ed remembers “overnight, I became the application port guru. People were e-mailing me from all over the world, trying to get their applications ported over.” He also counts driver and kernel work among his larger contributions.

Unfortunately, nowadays many of those moving most quickly around the Linux community are, for lack of a better word, the carpetbaggers. To those people, Ed shows very little regard, admitting that

> I think [Linux] would've survived regardless—there are a lot of operating systems around that haven't received the publicity that Linux has, and are doing quite well ... I think the popularity of Linux can be a two-edged sword.

Ed is still involved with computers and, more importantly, still involved with Linux. He says, “I'm involved in two projects, one to bring a port of Linux to a PowerPC platform for an embedded ATM controller, and two to develop a new server-side scripting language for the Web.”

And outside of computers?

> I backpack, and am also involved in ham radio. I don't think I've wasted my life. Working with computers has produced a lot of personal satisfaction for me. I've written pager notification systems for severe weather alerts, and I'd like to think that software has saved a life or two along the way.
>
> Ed Carp's e-mail address is erc@pobox.com.

Alan Cox

Among the Who's Who of the Linux Kernel, Alan Cox is probably one of the bigger Whos in the bunch. From his work on the Linux networking code to his current role as maintainer of the stable kernel releases, there are few who have meant as much to Linux as Alan.

> I like the flexibility and the control of free software. Most of my experiences with proprietary software have been either getting screwed as a user or being part of a larger company that had to threaten its suppliers with lawsuits to get service.

Somewhat provocative, but a familiar cheer and lament from many who have spent a goodly amount of time with Linux and open-source software.

Alan was actually working on ideas for his own operating system when his interest in Linux first began.

> I had pondered getting a decent PC since the Amiga was getting a bit long in the tooth. 386BSD came out, and it looked like finally there was an OS worth running on x86 hardware. Linux came out about the same time, but didn't need an FPU, so I started running Linux.

As one of the operating system's true progenitors, Alan is well aware of the importance of the GNU project to the development and maturation of Linux.

> In fact, in many ways Linux exists because GNU chose to pursue the HURD rather than using UZI as their UNIX OS core ... GNU/Linux is perhaps overstating it, but ignoring the FSF (Free Software Foundation) contribution is even worse ... It's really x11/BSD/GNU/.../Linux.

Alan now works for Red Hat, the poster boy and problem child for many in and around the Linux community. He still has time, however, to hack free software at home, as well as visit friends and colleagues while traveling to conferences and trade shows. While there are many more people who have contributed to Linux than could ever fit under one roof, Alan seems to have made his peace working for "a vendor."

> I get regular mail from people trying to find Linux-aware folks to hire. I think those who wrote code for fun have plenty of opportunity to reap rewards. Even when I wasn't working for Red Hat, it didn't bother me. I wrote it for fun, and the fact that people found it useful was a greater reward than money.
>
> Alan Cox's e-mail address is alan@lxorguk.ukuu.org.uk.

Laurence Culhane

Among those who made major contributions to the Linux kernel yet moved on to a relatively hack-free lifestyle, Laurence Culhane is one of those who stands out. A radio presenter for the BBC when he first encountered Linux, today Laurence is a senior BBC journalist. But in between then and now, he enjoyed several heady years in the thick of Linux kernel hacking. He recalls,

> It was fun. At first I didn't realize how revolutionary the idea was—it seemed so natural. It wasn't perfect, and with only limited free time, I found it quite hard to keep up with the demands for improvements to what had initially been just a quick hack to keep me connected.

Like many others, including Linus Torvalds himself, Laurence's Linux hack sprang from "purely selfish reasons." As he tells the story,

> I wanted Usenet and e-mail, and for that I needed SLIP/PPP. Neither had been written at the time, so I went and looked at the RFC's and wrote something that gave me adequate SLIP access.

Laurence wrote the original alpha SLIP code for the Linux kernel "and sent Linus the odd patch when I couldn't get other code to port and it looked like a kernel issue." Given Laurence's need-based arrival in the world of Linux and open-source software, it is little surprise that much of the philosophy behind

the Linux movement was initially lost on him: “I hadn't thought about free software, free speech or anything at that point. I just wanted Linux to work.”

What attracted him to Linux? “The fact that it was free was the first reason I looked at it,” Laurence admits. “I'd just left the university, had no money and certainly couldn't afford thousands of pounds [for] a commercial UNIX license.” It was at the university where Laurence got his first taste of a major UNIX system. Having built his first s-100 z-80A system at the age of 15, he “fell in love with BSD UNIX” while at the university, “found a 32016 S-100 CPU and MINIX and never left.”

Although Laurence no longer considers himself a Linux developer, he is still a regular Linux user and tries to keep up with the latest developments in Linux in general and the kernel in specific. While he thinks the current popularity surrounding Linux is “great,” he thinks a little reserve is probably a good idea. He says,

> I think it's important that people don't get overzealous about Linux ... I'm a passionate fan; I had never even used an MS product until 1998 when work dictated that I did. [But] I'm suggesting that my dad have a dual-boot Linux-Windows machine, with a Mac emulator under Linux because having access to all three operating systems is the right solution for his job.

Laurence Culhane's e-mail address is laurence.culhane@bbc.co.uk.

Thomas Dunbar

Another mathematician who spent some time with the Linux OS during its formative years is Thomas Dunbar. “Getting TeX/METAFONT running on Linux” was his most significant contribution to the operating system. Thomas had been using MINIX and doing technical typesetting work with TeX when he first started thinking of Linux as a way to move from his life as a math professor into “something computer-related.” Says Thomas,

> I needed a low-cost, programmer-friendly OS that could use minimal PC hardware. There were no barriers, neither technical nor social, to helping with Linux development.

Like many kernel hackers, while working over the Internet was both a fun and necessary part of working on Linux, Thomas never saw the Internet part of his work as particularly revolutionary—although it did play a positive role in promoting “family values.” As Thomas tells it,

> Linux coming along right when the Net was opening up was very fortunate. I did not feel this myself too much, as I already had professional contacts. However, I know this was very significant for my son, Daniel, who was just entering [his] teens when "we" started using Linux. The resulting accessibility of expert developers provided an alternative to the normal college, job training ...

Thomas still uses Linux on his desktop (as does his secretary), although he hasn't done any development work since his early TeX/METAFONT days. He sees commercialism as something that has, for the most part, helped increase the popularity and use of Linux—yet he doesn't believe Linux will become the "dominant desktop."

Currently, Thomas is senior DBA for the Warehousing group at Virginia Tech and also CEO of a little WebCyS shop, diads.com. Says Thomas,

> My work is almost exclusively Oracle-related with databases running on Sun/Solaris (though that is migrating a bit to Linux) with Linux being a convenient client platform for system administration and database administration of the systems.

What is the most important thing to him about Linux today? "Keep it fun and sociable, wherever that leads," Thomas says. "The more popular, the better, in my opinion."

Thomas Dunbar's e-mail address is tdunbar@vt.edu.

Bjorn Ekwall

Bjorn Ekwall, whose contributions to the Linux kernel include D-Link drivers, likes to make sure that some of the people who helped him discover and solve problems aren't forgotten. Hackers like Joshua Kopper and Jacques Gelinas, to name a few, are among the first Bjorn mentions.

> There were a lot of other very skilled hackers involved ... and I tried to make sure that all got credits for their contributions. I definitely learned a lot from these guys.

Bjorn first got involved in Linux through his devotion to UNIX in the late 1970s. Having worked on UNIX machines as a co-sysadmin, Bjorn had little time for the Apples and PCs that started shipping ("just toys," he called them), but because proprietary Unices did not offer the freedom or the source code Bjorn wanted, he had to rely on merely collecting and building the parts of his very own UNIX machine, "with full source."

It wasn't until 1992 that, while looking through comp.sources.unix postings on Usenet, Bjorn first saw mention of Linux. After a bit of independent research, he settled on an early Slackware distribution. “I think it was with the 0.99.3 kernel,” Bjorn says. But his reaction was epiphanic. “It had all the sources! It even had X11! It fitted neatly into my new 386SX/25 laptop with 5MB of memory!” says Bjorn. “It was close enough to UNIX for me to give in completely!”

And “give in” he did. After developing a D-Link driver for the Linux kernel to allow him to network with his laptop, and a few more projects dealing with kernel downloadable modules, Bjorn sent snapshots of his patches to Linus. “He answered by putting the whole package up for FTP together with the official kernel sources,” Bjorn says. “I admit that I was a bit flattered. A lot, actually. Suddenly I had been promoted to an official kernel developer/maintainer!”

The responses from fellow Linux hackers was something that made a serious impression on Bjorn. He says,

> the openness in the acceptance of new ideas and the ease of getting quick and high-quality feedback is definitely the most important thing about Linux, as far as I'm concerned. The basic rule, “show me the code”, is key, since it keeps in check those who have only opinions and no solutions.

All the same, much of the Linux politicking tends to leave him cold. Asked about commercialization, Bjorn says he has “no problems whatsoever” with people making money from Linux, “as long as Linux stays open, which it will,” he adds. Asked about Microsoft, Bjorn says frankly that he doesn't care:

> I'm only interested in getting access to an environment that fills my needs, which is what my Linux-based system does. If I need something completely new in my environment, then I will build it. If that is useful for other people, then that's a nice side effect.

But when asked about life outside of computers ... now that's a different story.

> There is definitely a world outside of computers, and I try to enjoy it as much as possible ... I do “have a life,” which includes my two daughters, now aged 9 and 12. We have a lot of fun—when I'm not working, that is.
>
> Bjorn Ekwall's e-mail address is bjorn@blox.se.

Drew Eckhardt

For some, Linux represented the on-ramp to a life of hacking on the open-source operating system. For others, Linux represented a momentary opportunity to explore interesting, non-trivial software development work. It was in this latter group that Drew Eckhardt found himself as an 18-year-old CS student at the University of Colorado. Like other Linux hackers, Drew was not satisfied with Bill Jolitz's BSD work, and the attraction to a freely redistributable UNIX system proved irresistible. He told me,

> I wanted to run some free UNIX on my hardware. Since I didn't like what Bill Jolitz was doing, that meant Linux.

Drew's first problems with his new Linux system led to his first contribution to Linux development. “I was too impatient to wait for someone to fix these problems (boot blocks that didn't work, disk driver problems), and solutions ... weren't too difficult,” he says. “Farther on, I continued to contribute to the Linux kernel because it was fun.”

Drew spent most of his time as a “Linux developer” working on the SCSI subsystem. But he is no longer involved in the development end of Linux. “Developing for the Linux kernel and user lands would be too close to what I do at work,” he says. The little UNIX hacking Drew has done on his own recently has tended to be FreeBSD.

While Drew emphasizes that the size of the Linux community of hackers was one of the best things about it, he doesn't think there is anything too revolutionary about the way Linux was developed. He suggests,

> In hindsight, the development effort wasn't too different from commercial environments where developers hide in their offices, work on some subsystem and release the code as certain functions are completed.

Drew may not play much of a role in future Linux development (he is a software engineer for a company that builds digital video servers for broadcast and post-production). But his thoughts on the future of proprietary software vs. open-source systems do reveal a future for Linux. He says,

> In niche markets, we'll always have proprietary software because those markets can't or won't fund new products, and software companies can't guarantee they'll sell the support needed to pay for development after the fact. In the general consumer market, its days may be numbered ... buying shrink-wrapped proprietary software is a bit silly when you

> can get the same software on a CD-Recordable for a dollar.
>
> Drew Eckhardt's e-mail address is drew@poohsticks.org.

Rik Faith

Some people find Linux, stay for awhile and then part ways. And others? Well, for some, when it comes to Linux, once you hack, you'll never go back.

Rik Faith is currently working with Precision Insight and, as such, gets to spend all of his work time “using and improving” Linux. Says Rik,

> The popularity of Linux and the willingness of vendors to pay for Linux improvements (both in the kernel and in user space) have enabled me to find what is very nearly my ideal job: I can work at home, use Linux all the time, and get paid for improving Linux and XFree86.

Rik first discovered Linux while toiling away in graduate school. He had been working on his Ph.D when he heard “rumors about a free UNIX” in late 1991. All the same, it wasn't until spring of the following year that Rik actually downloaded the source code and booted it up. As Rik remembers, “it booted fine from floppy and was able to see my old 40MB drive, but it didn't support my Future Domain SCSI controller.”

And this is when Rik Faith's inner penguin started singing.

> I ordered a manual for the Future Domain chip set, and as soon as finals were over, I started to write a SCSI device driver. After about three days, I was convinced that this was too difficult for me. But on the fourth day, I had a working SCSI driver!

And by the end of the month, Rik's hack was fully interrupt-driven and ready for Linux 0.97.

In addition to his work on the Future Domain SCSI driver, Rik also worked on the APM driver, and later did kernel work on the Direct Rendering Interface (DRI) as an engineer with Precision Insight. And like some of the other original kernel hackers, Rik has been involved in a number of non-kernel Linux projects. These include maintenance of the util-linux collection and coordination of the man page project—both of which have since been taken over by Andries Brouwer, another of the original Linux hackers. Rik even worked on his own Linux distribution, BOGUS Linux, with colleagues Kevin Martin and Doug Hoffman. “The BOGUS Linux release was the first Linux distribution to use the

`pristine source plus patches' paradigm that is now familiar to all RPM users," Rik notes.

All this nonstop Linux work has kept Rik exceptionally busy over the years. In fact, he says,

> I found that I had to cut back on my Linux work for a few years while I finished my Ph.D and started a family —my wife, Melissa, and I have two daughters, Rhiannon (4 years) and Selena (7 months) ... For fun, I spend time with my wife, play with my kids, and work on free software to format, search and serve human-language dictionaries.

Rik Faith's e-mail address is faith@alephnull.com. Visit his work with "human-language dictionaries" at http://www.dict.org/.

Jeremy Fitzhardinge

Why did the hacker hack the kernel? According to Jeremy Fitzhardinge,

> I started hacking it for the reason that everyone hacked on it then: there was a lot of stuff it didn't do right, and there were things I wanted it to do for various programming projects.

In the beginning, Jeremy was working at his first "real job" when he got deep into kernel hacking. He says,

> I was also looking at relatively obscure research operating systems (Amoeba, Sprite) and wondering whether I could run one at home. Then a friend showed me Linux, and I was amazed at how concise it was compared to, say, SVR4.

Jeremy, whose contributions to the Linux kernel include work on file systems and VM (virtual machine), is currently working on more "stuff it didn't do right," such as autofs, the Linux automounter that he has been trying to improve. He is also one of those hackers who has been fortunate enough to make a living hacking Linux. Right now, most of his work is in embedded systems, getting Linux to boot out of flash memory on a "reasonably powerful PPC-based server the size of a CD-ROM drive."

And, as might be expected, Jeremy is thrilled with both the prospect of commercial applications being written for Linux, as well as the overall popular success of Linux these days. If anything, however, Jeremy sees little sense in fixating on competing with Microsoft. He tells us,

> The number-one threat [to Linux] is thinking that competition with Windows is important, or indeed, that all the commercial interest is important to Linux.

Let Linux be Linux, Jeremy seems to suggest, very much as many of his hacker colleagues continue to urge:

> There's still a way to go before people will happily sit down to a Linux machine and do work, because the desktop applications are not there yet. I like the fact that there's lots of different efforts, but they should all go to some effort to keep their file formats interchangeable wherever possible.

All the same, he thinks Linux does undermine many of the pretensions of shrink-wrapped, proprietary software. “I think people will become wary about buying closed-source programs as the quality of open source improves.” Furthermore, Jeremy believes that the expectations open-source customers have will make closed-source vendors more accountable.

Philosophically speaking, Jeremy is among those “kernel forefathers” who places high value on the GNU Project, even though his support for the GNU Project is more on the practical side. “Without GNU, we'd be stuck without a serious compiler to base everything on, and also be without many of those programs which make the UNIX experience,” he says.

Jeremy Fitzhardinge's e-mail address is jeremy@goop.org.

Philip Gladstone

Philip was interested in making Linux a good platform for timekeeping. He added the kernel phase locked loop and “fixed a bunch of problems with timekeeping.” He continues to assist in this area of development, but his “principal area is now bug-fixing obscure conditions.”

Linux was not Philip's first experience with developing software via the Internet. He had done so previously with Tom Lane. This work produced the initial version of the IJG (Independent JPEG Group) JPEG library. The recent mission of the Mars Pathfinder used JPEG encoding, but Philip has not yet been able to determine if this included his work.

In 1991, Linux found him working as a consultant for a large New York City-based bank. Philip was concerned with connecting the company to the Internet. He relates it best:

> We needed to build a system to act as the name server. Somehow I found out about Linux, and I built a

> system on a 386 running 0.99pl15 and bind. It served as the external DNS server for many years—only rebooting after power failures.

Philip “can't say anything” about his new job, where he works for a new startup in the Network Security field. He still runs Linux on his laptop and home systems, but home life allows for only the occasional bug fix. He adds, “my principal contributions [to Linux] are now work-related. There are times when I produce something at work that can be given back to the community.”

He still views Linux as a good platform for “certain types of solutions.” Having more support for Linux from hardware manufacturers on the driver end is necessary. “It's not realistic to rely on the free software community to produce drivers for everything.” Philip doesn't see Linux taking the desktop in the next three years, citing “no consistency in user interface style” as one problem. He does feel that Linux should be able to hold the server market, as long as “the management tools are improved.”

Is there a life outside computing? Philip says, “Very much so. If you never have children, then you will miss out on something much more important than computers.” He is married, has a daughter and another child is on the way.

Philip Gladstone can be reached at philip@gladstonefamily.net.

Dirk Hohndel

Dirk has been involved with Linux since the early days. He was 24 and a student of mathematics/computer science at the University of Wurzburg in Germany. Working as a system administrator within the university, he “found postings from Linus in the comp.os.minix group, where he talked about a project of his.” Dirk wanted a UNIX operating system for his own machine. “FreeBSD didn't exist and 386BSD wasn't an option,” says Hohndel. Linux was the best choice, so he began to develop.

With an interest in memory management and adding hardware support, he helped with the first Ethernet drivers (for the WD8003) and the first SCSI support (ST01). “Soon thereafter,” he says, “I became involved in the first

implementation of shared libraries (based on jump tables) and applied that for the XFree86 shared libraries." Dirk can recall an e-mail exchange with Linus "where we joked that at some time, it might be possible to run X on Linux." Since then, he has continued his work with the XFree86 project, currently holding the title of Vice President.

Dirk was recently named CTO of SuSE Linux AG. This means he spends much of his time managing projects and not so much on programming. "Working on XFree86 is what I do for fun," he says with a smile. Helping Linux continue to grow is an overall goal. "Making Linux easier to use for the end user" is one area for Linux developers to place focus on. Dirk doesn't see this as "dumbing down," as it in no way subtracts from the power and flexibility of the operating system.

Claiming to have a life outside computers, Dirk said, "Having friends that know nothing about computers and get bored when you talk about computers really helps." Do such people exist?

Dirk Hohndel can be reached via e-mail at hohndel@suse.com.

Nick Holloway

Nick Holloway doesn't think the beginnings of Linux were all that revolutionary when it started out. As Nick recalls,

> I have seen it in action with the various source newsgroups (alt.sources, comp.sources.unix, comp.sources.misc), where I could make changes, submit them back to the author, and see them in the next release. Initially, Linux wasn't all that different. It was just an OS kernel, rather than an application. It just grew to be a much larger scale.

Nick considers his contributions to Linux to be relatively modest—as do many original kernel hackers.

> I was interested in the areas that I needed to work for me. I contributed patches to libc4 when I found problems ... I contributed tab expansion for the tty layer in the kernel when I wanted to use a dumb terminal that couldn't handle hardware tabs. However, these days, my involvement normally is restricted to tracking the Linux kernel mailing list and browsing the patches. I'll submit minor patches from time to time, but I am not a mainstream contributor.

As a Ph.D. student at the University of Warwick, Nick first heard about Linux through Usenet. "I immediately subscribed to alt.os.linux so I could read more.

In early 1993, I bought a machine specifically to run Linux." Nick was one of the many Linux hackers who was weaned on UNIX, having used both BSD and the SunOS "almost exclusively" since starting at the university in 1985. The problem was that he wanted a home computer *and* he wanted to run UNIX. "When Linux became available, it was the obvious choice to me," he says. "It had enough to get started and be usable, but there was plenty of scope for being able to contribute to the development."

This best-of-both-worlds thinking carries over to Nick's opinion of Linux's present-day situation. The open-source operating system's exceptional popularity, he thinks, has definitely helped quicken the pace of development, guessing that Linux might have remained "a hacker's plaything" otherwise. As such, Nick believes there is a place for commercial applications being written for Linux. He says,

> Just because the OS and many of the standard applications are free doesn't mean they all have to be. If a company has to invest in producing an application for Linux, then they have the right to charge for it.

In fact, as far as Nick is concerned, such so-called profiteering can actually end up helping the Linux development community. "For example, Red Hat and SuSE are in the position to employ important hackers, which means [hackers] don't suffer from real work getting in the way of their Linux work."

Which is something Nick knows all too well. Currently employed in "the development of business-to-business e-commerce solutions," Nick spends his work time with Windows NT and Solaris. All the same, he says, it's not so bad. "It allows me to separate work and play in a clean way."

Nick Holloway's e-mail address is Nick.Holloway@alfie.demon.co.uk.

Rob Hooft

In typical open-source development fashion, Rob Hooft began hacking on Linux "because I could". There were limitations to the 0.95c++ kernel (the first one he installed at home). Rob says, "0.95c++ was only 200KB in .Z form" and it did hang on his home machine. So, he jumped into the kernel in order to

understand the driver and implement modifications. This was how Rob became a Linux contributor.

> I improved the OPL3 sound code in the kernel sound driver; changed the floppy formatting routines to use sector shifting; and helped develop some shared libraries.

Linux had little in the way of libraries then, but as we all know, Linux development grew because of what wasn't there.

When Rob first encountered Linux, he was at a UNIX users group meeting of the Dutch Hobby Club. Linux was the only true free UNIX. Ignoring the limitations, he “decided that Linux was what I had been waiting for, and I bought a computer (my first x86 after a Z80) specifically to run it.” He had a lot of faith, seeing that Linux was equipped with “only 64 processes, with 64MB virtual address space each; only four partitions on a hard disk (maximum 64MB each); init/getty/login (IGL) were barely completed; and no X.” One of the true wonders of the Linux world comes from the belief early developers had in the potential of Linux.

Rob is now a programmer for Nonius BV, which makes “machines that are used for crystal structure analysis. The control software is written almost completely in Python on a Linux machine, using Tcl/Tk GUI.” He is using only free software to create a commercial application. Rob made the shift from academia to industry because of his wife and son, and a little thing called job security. Not surprisingly, he doesn't have a problem with, as he puts it, “smart brains getting rich, even if they're not programmers.”

Though he writes commercial software, Rob tries to make certain modules freely available, but

> only if they are generally applicable. If I make the whole lot open source, the only one studying it would be our competitors. My competitors are not feeding my family.

Fair enough. Rob Hooft can be reached at rob@hooft.net.

Olaf Kirch

Olaf discovered Linux while writing his master's thesis in mathematics. He says, “Linux was so cool that I ended up spending a lot more time on it than on gnawing pencils over my theorems.” He did finish the thesis eventually and went on to write the *The Linux Network Administrators' Guide* (first published in 1993). This was his first Linux project, and it has since gone to paperback and no doubt sits on many a system administrator's desk.

Attracted to Linux by the lack of an “initial” hierarchy, Olaf worked with Jeff Uphoff on the first Linux security list, then turned his efforts toward Linux NFS implementation. He has maintained the user space NFS daemon for the past five years, but has “toned down” his overall involvement to spend more time with his wife Maren and their 18-month-old daughter.

Looking back on the early days, Olaf was struck by “the cooperative spirit that reinforced itself.” Nothing has changed there. The community has always been the driving force behind Linux. The Internet has always helped to connect to the community. Back then, Olaf relates, “it [the Internet] felt like giving money-grubbing UNIX vendors the finger.” He credits *not* wanting to pay hundreds of dollars for inferior software as a key factor that drove early Linux development.

As for the future of Linux, Olaf is hopeful that the major Linux distributors will adopt the Linux Standard Base definition. Olaf sees platform differences between the various distributions as an increasing obstacle to software vendors entering the market. He believes that “once LSB is finished and accepted, there will be a common base platform for porting commercial software to Linux, and we will experience a surge in new software being made available on Linux.”

He currently works for Caldera, doing “security and a lot of network stuff, as well as developing an admin tool framework.” Working for Caldera affords him an opportunity he did not dream of eight years ago: being paid to work on Linux.

Olaf can be reached at okir@caldera.de.

Ian Kluft

Ian Kluft has done a lot for Linux and open-source software. He founded sbay.org in 1993, which he calls “a communications geek group in Silicon Valley.” He is still the group's coordinator and notes that Linux has been the backbone of the group's infrastructure since the beginning. Ian also helped with the Usenet Volunteer Votetakers. He contributed a module (mod_mime_magic) to Apache and helped start the Apache JServ (Java servlet engine) project.

His work with Linux was born out of time spent in Amdahl's mainframe UNIX lab in 1992, where he first discovered Linux. He says, “I had been involved with maintaining the **smail** mail transport agent on Amdahl's mainframes.” He noticed that smail was not yet ported to Linux. He did the port on a 486, then passed his work on to the smail maintainers. “Little did I know at the time, I had made the first Linux e-mail server.” He continued his work with smail by maintaining the Linux binaries on Sunsite, an early Linux FTP site. He maintained smail for Debian and Slackware until mid-1995, when “it looked like the Linux distributions were able to handle e-mail servers on their own.”

Ian was attracted to Linux because he didn't feel limited by a commercial organization. Like many, he didn't want to pay SCO $800 for a two-user UNIX license. After a few weeks, he had Linux running on a 386. The real draw was that there was no one telling him what he could or could not do with *his* computer. “Any code I could write, I could run. Anyone competent enough to be the system administrator of their own server had uncommon computing power in their homes.” Like the rest of us, the current success of Linux astounds Ian and he certainly “never expected to end up with Linux stock worth about a year's salary.”

In 1995, Ian left Amdahl for a smaller, lesser-known company called Cisco Systems. He is currently a software engineer in Cisco's IOS technologies division. Outside the computing/Linux world, Ian involves himself with the West Valley Amateur Radio Association, serving a term as president. He has a strong interest in amateur rocketry and works with other rocketry hobbyists “on using a real-time variant of Linux as an on-board flight control system for a suborbital

amateur rocket." He spent a week in the Nevada desert in March to help a group from Sacramento attempt the first amateur rocket launch to space.

Ian can be reached via e-mail at ikluft@thunder.sbay.org.

Michael K. Johnson

Michael was introduced to UNIX during the first week of his freshman year at St. Olaf College. He was "deeply impressed by the communication potential of a true multi-tasking and multi-user system." He was primarily attracted to UNIX, but his student budget sent him looking for another alternative. BSD386 wasn't available, and there was no free software alternative. He tried Coherent, "but they didn't deliver on their promises [SCSI drivers, networking, etc.] and weren't interesting as a result."

After the release of kernel 0.02, Michael downloaded Linux and placed an order for an IDE hard drive to run it. "I was excited to see that my dream of a home-built operating system was viable." From there, he read the entire kernel source code in an attempt at understanding the user space. He has worked on many aspects of Linux development, but focused mainly on user space, "specializing in system component interface and integration." Michael helped Matt Welsh with the Linux Documentation Project and worked on writing procps and the parallel port device driver, as well.

It is interesting to hear Michael talk about the early development of Linux. He admits that "...there was no way you could call me a hacker. I was quite clueless—as was, in some ways, Linus when he started." The early developers of Linux weren't always experienced programmers. Many learned as they went, sharing their knowledge with others. He says, "we developed a community that worked despite its imperfections ... and didn't waste time pondering the idea that we might be making history."

He isn't upset with the commercialization of Linux, saying it "has made it possible for me to put Linux on my father's computer." This is something the early developers probably didn't foresee. At least it wasn't *the* reason they volunteered their efforts. Michael had no idea he would be able to make a

living by working on Linux. Like many, he worked on Linux to learn, and as he says, “that learning was sufficient compensation for my volunteered efforts.”

Michael currently works at Red Hat, which he says provides a good balance between his desire to work on Linux and the need to make money. He was an early editor of *Linux Journal* and co-authored *Linux Application Development* with Erik Troan. Michael considers this to be his “biggest indirect contribution to Linux development”. He can be reached via e-mail at johnsonm@redhat.com.

Bas Laarhoven

Bas wrote the ftape driver. A Linux driver did not exist, so Bas decided to write one. His reason: “because I needed it and thought it might be useful for others.” This is how open-source software gets developed. His continued work with Linux produced the code that would become the kernel modules package. The package allowed driver development and debugging without constantly having to reboot. As Bas says, “It also allowed the user of the ftape module to load it only when needed, keeping kernel memory usage as low as possible.” Though his real job forced him to “pass the ftape driver to a new maintainer,” his modules code was eventually integrated into the kernel tree.

In early 1993, Bas “found a complete `free UNIX clone' on a local BBS.” It took him a few weeks to download the kernel, but the result was the ability to use UUCP to retrieve information from the Internet, including e-mail. “That was part of the thrill: development via the Internet was fascinating.” Bas credits the Internet with bringing together highly talented and motivated people, most of whom were working to “contribute to and create the most perfect system.” His experience was far from unique.

Bas still works at the same company as he did back then, but is contemplating ways to combine his family life and make money developing Linux. This shouldn't be too difficult for someone with his credentials and commitment to open-source philosophy. However, he is concerned that the large sums of money being tossed around in the name of Linux will change everything. He worries that the success of Linux might pose the biggest threat. “Way too much

money is involved, and I'm not sure that Linux is ready to replace the Microsoft-based environments yet," he says. Bas would like developers to place more focus on making Linux easier to install and on making the switch from Windows more transparent.

When asked if there is a world outside computers, Bas replied, "Tell me more about this world." He lives in the Netherlands with his wife and 18-month-old son. He can be reached at bas.laarhoven@home.nl.

Warner Losh

Warner Losh isn't so enamored with Linux these days. He says, "I've grown beyond Linux. It was a cool hack once, but Linux doesn't live up to its hype." Currently, Warner is a software engineer working on FreeBSD. "I have a high profile there," he says, "which gives me chances to work on consulting projects to augment my income." He is a senior committer to FreeBSD along with serving as its security officer.

When his opinion of Linux was higher, Warner worked to get his company's C++ toolkit and GUI open sourced. He also worked on porting Linux to the ARC-based RISC machine, which was built around the MIPS processor at a time when Microsoft was touting the Intel processor. He was also involved with the early efforts to bring Linux up on the Windows CE devices. His Linux days are basically over, though. He thinks Linux's development model is horrible. "I stopped using Linux on my Intel machine because it took so much time to keep up to date. Download a new kernel, and you will need a new libc. But that libc will also need a new gcc, and the new gcc needs new bitutils. It all got very hard to keep track of."

Now, Warner prefers the FreeBSD development model, "where you get the entire tree from one place ... which makes it easier to contribute back to FreeBSD faster." He thinks the GPL is "evil," preferring the BSD license "because it encourages cooperation." Cooperation between developers made Linux what it is today. Warner refers to the "millions of random monkeys writing" for Linux as the best thing about Linux. However, he adds "the worst thing is that there's little filtering of their code or making the architecture coherent or stable over the long term." As for the future, if Linux is to continue to succeed, he says, "it needs to organize its chaos."

Warner Losh can be reached via e-mail at imp@village.org.

H.J. Lu

H.J. Lu first discovered Linux almost a decade ago in 1990, while he was a student looking for a "decent OS" for his 16MHz 386sx with only 4MB RAM—a

common theme among many of our original kernel hackers. Not a big fan of Windows, and unable to get his hands on BSD, H.J. not only started running Linux, but soon got involved with Linux development. “Many things didn't work and I needed them,” he says.

As a Linux developer, H.J. worked on the kernel's C library, binutils and gcc, the latter a project he had been working on before he even learned about Linux. Similarly, H.J. had learned about the GNU project—which he calls “very crucial to Linux”—before starting on Linux, as well.

Without a doubt, having access to the source code is what made (and makes) Linux worthwhile to H.J. “We can fix it if it doesn't work,” he emphasizes. At the same time, H.J. is undaunted by the increasing commercial interest in Linux. “I think Linux will attract commercial developers who are threatened by Microsoft and want to make money on Linux,” he says. Given the fact that H.J. works for VA Linux Systems, it is perhaps no surprise that he is less intimidated by commercial interests than others may be.

> I don't mind that people profit from my work. I do it because I enjoy it—it is not bad to get paid to do what you enjoy.

Where does Linux need to go from here? H.J. has a couple of areas he would like to see worked on further, including a journaling file system, “decent NFS” and improved VM and networking. As for the Linux-on-the-desktop battle, he is convinced that more desktop applications will be important in winning over more end-user converts, whether the applications are commercial or free. “But the basic stuff,” he adds, “should be free.”

H. J. Lu's e-mail address is hjl@valinux.com.

James MacLean

James MacLean lives in Nova Scotia and works for the Department of Education as manager of technical services. He says,

> We use Linux all over the place at work. It was accepted early on for anything we could get it to do.

This makes for a good balance between work and Linux. Plus, he gets paid. Not surprisingly, then, James believes “all the financial activity is part of what makes Linux fun to be a part of.” He doesn't resent or fear the money influence. Instead, he welcomes the commercialization of Linux, seeing it as “an honest way to support Linux.” James says, “the biggest addition [needed for Linux] would be more applications that are currently supported on Windows.”

He expects Linux to grow on the desktop, and believes that “over time, it will leave fewer reasons to *not* deploy it as a desktop solution.” The only thing he sees that could ultimately hurt Linux development would be if the leadership in Linux diminished. In case we forgot, James reminds us that “taking over the world” is still the goal, and he thinks it is highly unlikely that the “people and parts that make up the heart of Linux” will curtail their devotion to its development.

James discovered Linux through a friend who had seen mention of a “free UNIX-type OS” in *Byte* magazine. Not quite thirty and newly employed by the government, he worked to automate the mainframe connection in order to collect data. He was “trying to make use of OS/2 2.0 on a 4MB 386DX33 ... but missed the flexibility of UNIX.” OS/2 was too slow, and Windows wasn't production-ready: “It crashed too much with the applications I tried to use.” It's reassuring to know that some things never change. James eventually found Linux.

He doesn't have much time these days to work on Linux outside of the office, but he does appreciate all the effort being put forth by others. James does custom things around the office to meet the specific needs of his section. This work generally does not find its way outside the office, but someday it might. He points out,

> It used to be that even in a crude form you could put things out for others, but now I get shy, because if it will not work out-of-the-box, then I have to be ready for storms of e-mails.

James can be reached at macleajb@trademart-1.ednet.ns.ca.

Kai Makisara

Kai Makisara didn't begin working on Linux *because* he is Finnish, but it did seem *proper*. “I was using UNIX systems at work and wanted to be able to do some research work at home,” he explains. Linux supported most of his hardware, so he chose it over the BSD variants and Windows, which he could not run on his hardware without major changes.

Kai first saw references to Linux in network news. This was in 1992, and he was 38 years old. At the time, he was working on remote-sensing methods at the Technical Research Centre of Finland. Coming from a research organization, he found development via the Internet to be “natural and not revolutionary.” He had been using “the network to exchange messages, software and data for a long time.”

Kai began working on an area of Linux that related best to his background in computer science. He says, “I had used SCSI devices in our workstations, and tapes were/are an essential part of remote sensing work.” So, making a SCSI tape driver was a natural choice. Kai still maintains the SCSI tape driver and posts the occasional fix.

“The biggest threat to Linux,” in Kai's opinion, “is that all key people working on Linux lose interest.” This has always been a bit of a problem, but thankfully there has been no real shortage of developers willing to contribute. He doesn't “think that Linux should compete with Microsoft in the fields where they are strongest.” Kai finds “systems that enable people to process information anywhere” are much more interesting than the traditional desktop. “I have never accepted the attitude that work can be done only [by] sitting on one's chair facing one's personal computer.”

“Continued development is necessary for Linux's survival,” he continues. The current popularity of Linux makes him happy, as it shows that good software can be successful without being backed by large sums of money. But he does point out that “if better and affordable solutions for my computing needs emerge, I may jump ship.”

Kai continues to work on remote sensing research at the Finnish Forest Research Institute and can be reached at makisara@metla.fi.

John Martin

Looking back on the meteoric rise of Linux over the past few years, some of our original kernel hackers have expressed surprise, enthusiasm or a cautious optimism over the distance the open-source operating system has traveled, from Internet hack to Wall Street/Silicon Valley darling. But in and among the many comments we've heard a slight tone of nostalgia for the old days, typified by the passing comment from one of those original kernel hackers, John Martin.

> It might have been more fun within a small community. With the Internet early on, there was a sufficiently large community to sustain itself, while it was not viewed as threatening to the powerful. [However,] once Linux got bigger, it may have been important to get very big, very quickly.

Today, John Martin is an independent consultant with “practically no time to call my own,” he says. But once upon a time, he was just another hacker looking for something interesting on which to ply his talents. “I was working with big iron, but was interested in learning about UNIX when I stumbled upon the Linux-activists list,” John says. He had a new Intel box that “would not run anything until I tried Linux, which worked immediately.” Worked, that is, except for a small file system problem that he was able to fix by way of a workaround developed by Stephen Tweedie. John was up and running within two hours of first trying to boot from the floppy. “I never looked back,” John says. “I wanted to do what little I could to put something back into the community.”

He calls the Linux activists “congenial and productive”. But, like many original kernel hackers, he didn't think he and his colleagues had embarked upon anything especially new, per se. “The challenge to the orthodoxy of large-scale software development was apparent long before Linux 1.0,” John notes. Similarly, John has little time for the financial pyrotechnics that have accompanied Linux's rising popularity as a potential “Windows killer,” nor is he particularly interested in issues of commercialization of Linux or what is often called Linux profiteering. John says,

> If commercial applications must be written, better they be written for Linux than for something else. Proprietary software and protocols are evil for practical reasons. Linux and open software and protocols are good for practical reasons. It is because these practical reasons are profound that they are embodied in a philosophy.

A tremendous fan of the GPL, which John considers the most important thing about Linux (“the GPL and the Linux development community ... seem inseparable”), he is no less passionate about open-source software in general.

“Support only open source,” he says. “Unless one believes that the value of pi and the human genome should be patented, this is no place for equivocation.”

John Martin's e-mail address is jam@jamux.com.

Bill Metzenthen

Bill was a 46-year-old Ph.D. student when he first began “using Linux 0.97pl2 and submitting bug reports in August 1992.” He gave up the “research environment” of his job as an electronics engineer to study theoretical physics. This led him to the mathematics department at Monash University. At the time, Linux did the things he needed, while UNIX systems were too expensive for personal use. Bill says, “I was sick of the closed world of proprietary software.” This was the impetus for his work in free software. He went from using MS-DOS and Turbo-C to DJ Delorie's port of gcc (djgpp). Linux took him in the direction he wanted to go.

Bill's involvement with mathematics gave him the ability he needed to help improve the FPU (floating-point unit) emulator in djgpp. Not being able to afford a machine with an FPU provided him with “the motivation to work on an emulator.” Linux had poor support for FPU emulation, so Bill transferred his work to Linux and continued the development. He isn't too involved with kernel development these days, but still maintains the FPU emulator.

He sees Microsoft as the biggest threat to “World Domination”, but hopes the recent ruling against Microsoft will benefit the revolution. The commercialization of Linux is a necessary element to continued success, he points out. And what of profiteering from the volunteered work of others? “Profiteering is an inherent part of capitalism as it exists today. Greed is good”, he says.

As for the future of Linux, Bill believes the desktop interface will need to be dumbed down. He adds,

> The trick is to keep the dummies away from the underlying operating system so they can't burn themselves ... sometimes you have to put a fence around dangerous places to protect people. Unfortunately, some people are still perverse enough to blame you if they climb the fence and fall over the cliff.

Outside the computer world, Bill is secretary of the Federation of Victorian Walkers. He tries to spend time away from the screen, but rationalizes that “there are times when I become aware that I am probably devoting too much

time to some activity, but life is full of illusions and the sense of wasting time is probably just another illusion.”

Bill Metzenthen's e-mail address is bill@melbc.org.au.

Pauline Middlelink

Pauline lives in Enshede, Holland and runs her own business, specializing in writing software and Internet consulting. She is also on the board of an Internet Service Provider, IAF, there. Believe it or not, not all her projects are Linux, but most involve the computer. She has forgotten what the word “fun” means (just kidding) and is comfortable with the computer dominance of her life.

She discovered Linux through the free UNIX version, MINIX. At first, she contributed small fixes here and there, but after developing an interest in connecting all her computers to the Internet, she developed the IP Masquerade module—real programming by a real programmer. Her main development interest for Linux today is in video4linux, maintaining the Zoran driver in the kernel, and “some VCR projects.”

With work and personal life taking up much of her time, Pauline finds it harder to keep up with developments and says, “I am having some concerns about getting behind current developments. It's hard to know what is going on, and with the advent of the Internet and worldwide development, things can happen incredibly fast.”

Pauline feels the Linux community owes a lot to the GNU project and thinks it might be fun to have a distribution from them named “GNU/Linux”. That's Debian, isn't it? She is more interested in Linux remaining a good, robust OS than in seeing it become popular in the user world—“a difficult low-profile market that doesn't bring in new developers.”

When I asked her what Linux still needed in order to continue growing, she told me:

> Full-blown USB. USB devices are crawling out of the wall everywhere, and not supporting all of them would be a deadly sin. For Linux to succeed more would not depend on the kernel itself; we need a good desktop with good office tools which are accessible to Joe Average, who is still a drag-and-drop kind of guy. Linux top qualities are still in the development speed, and when looking at common usages, the server market. It makes a tremendous network/web/mail server with outstanding firewall capabilities, which only will get better in the upcoming 2.4 series.

Pauline's home page is at http://www.polyware.nl/~middelink/En/ and she can be reached via e-mail at middelink@polyware.nl.

Rick Miller

Rick was hooked on Linux from his first kernel compile. “The experience is difficult to describe,” he says. Rick was into Linux before the first free BSD. He was an engineer at an electric power utility and was looking for a UNIX-like system to run on his PC at work. Then he heard about Linux on Usenet (comp.os.coherent). He “decided to give it a try because it was free.” Rick admits,

> ... another draw was Linus' own personality. One guy was so pleased at the money [given to him] he saved using Linux that he started a fund to send money to Linus. Linus nearly refused, complaining almost belligerently that no one should expect anything in return for the money and that he would probably spend it all on pizza and beer.

This only fed the flame, as the community began to rally around Torvalds.

Rick contributed to Linux by maintaining a “registry of device allocations.” Say you were writing a device driver and had this question: “Where can I find out what major:minor numbers aren't being used so I can use them in my new driver?” Rick saw this as a necessary service. Linus agreed, and Rick became the maintainer.

Since the early days, Rick's “career shifted into more computer-related fields where [he] was obliged to write Windows software.” However, his current job, as a key software engineer at Merge Technologies, Inc., involves developing medical imaging products to run on Linux. “We're creating our own distribution to carry it and I get to dabble in the SCSI kernel code!” Good luck.

Rick isn't bothered by money infiltrating the Linux community, and he doesn't share the view of some that undeserving profiteers are making money at the expense of Linux developers.

> The people making big bucks ... are the ones who take the responsibility to support what they sell ... who market it and put the customer together with the product. That's business, whether you're selling free software or bottled water.

Rick fully supports the proliferation of applications for Linux, commercial or not.

In his spare time, Rick is married and likes making flutes from cheap plastic pipes. He can be reached at rmiller@merge.com.

Corey Minyard

The number one reason Corey Minyard became part of the Linux community was Linus' attitude. He told me,

> Linus had the right attitude to make it succeed. I knew it would be big from the beginning. I helped because when things don't work right, well, it annoys me. Plus, everything else cost too much or was lousy. Except for FreeBSD, which I didn't know existed. I'm not sure it even did exist at that point in time.

He began contributing, like many others, by sending in small code patches, then went on to write the CDU31A proprietary CD-ROM driver and work on the TCP stacks to find race conditions. Currently, he is “doing some work on the PowerPC code; I did a major restructure to make adding new platforms easier and I did a port to the Force Powercore board. I'm doing lots of little things for work.”

Corey is one of those lucky programmers who gets to use Linux at work. He is a “system architect at a major communications equipment supplier.” However, he also believes in having a life outside work and Linux, and has a wife and two children to prove it. He says,

> Of course, a world exists outside of computers. They shouldn't be our raison d'etre. I worship in church, take my children to the zoo, spend time with my wife, and do a host of other important things. If I had to

> choose between computers and those things, computers would lose. Thankfully I don't have to choose. I try to spend my time in front of a computer helping others or learning. Those things are not a waste of time.

Corey can be reached at minyard@acm.org.

Johan Myreen

Johan Myreen is an easy fellow to talk to, giving his opinions and thoughts completely without prodding. He chose to work on Linux, discovering it at version 0.11, because Microsoft Windows had its “roots in toy operating systems, running on toy microcomputers,” while UNIX's roots are in a “more powerful computing environment.” He gave us this quote about operating systems:

> Somebody once said that if you compare operating systems to hammers, Windows would be a colorful and pretty to look at “Fisher-Price” type of hammer, while UNIX is perhaps not so good-looking, but is a sturdy tool that gets the job done.

Johan was and is interested in device driver development, “writing the PS/2 mouse driver and minor hacking in the keyboard driver.” He still does the maintenance needed for these drivers. He adds,

> In the early days, I also wanted to add support for diacriticals in the keyboard driver, only to discover that Linus had written the keyboard driver in assembly language. So the first thing I did was convert the driver to C. This was a long time ago, before we had loadable key maps.

Asked about working with others over the Internet, Johan had this to say:

> It certainly has been exciting following how Linux has developed over the years. I can understand the experts were skeptical, because pulling through a

> software project this large over the Net hadn't been tried yet. The Internet was a revolution. Probably the skeptics had a traditional view of the process, and didn't take into account the open nature of the development process. Linux was this great job advertisement, essentially saying: “Just pick any task you find interesting. You can start straight away.” (Of course, you didn't get paid for working on Linux.) Anybody could volunteer and jump in and work on what was interesting—there was no bureaucracy. What further boosted development was that, since nobody told who to do what, there was some internal competition. You had to work hard if you wanted to get your code into the kernel, before somebody else beat you to it.
>
> Compare all this to the traditional software development process with its managers, software engineers and testers, many with an “I just work here” attitude. Now try to manage this with your programmers scattered all around the world. The skeptics may still be right...

Johan Myreen can be reached at jem@iki.fi.

Alessandro Rubini

Alessandro has been a part of the Linux community from almost the beginning. He wrote **gpm**, the mouse server for the Linux console, and we are all happy he did—what would we do without our mouse?

Attracted to Linux because of its freedom, he bought a computer just to be able to play with Linux. He started hacking the kernel with version 0.99.14 to get it to work with his oldest video board—a patch he considers “dirty” but necessary for him. He also contributed a one-line patch to kernel 1.0.6 to increase performance in low-memory situations. His primary programming interest is in drivers.

A long-time author for *LJ*, Alessandro's first article for us appeared in the September 1995 issue (#17) and dealt with writing mouse-sensitive applications. In the March 1996 issue, his articles for Kernel Korner began appearing, and he became one of our most popular writers, among both our

readers and us. His articles were always accurate and needed very little editing, even though English is not his native language. Would you believe he's Italian? He is, and a most dashing one at that, with a dark and brooding look in the pictures he sends us.

After completing a degree in electronic engineering, Alessandro decided to have a life outside Linux, so he got married and started a family. He has one child, and a second is expected. Even while devoting much of his time to family and work (writing free software), he managed to find the time to write *Linux Device Drivers* for O'Reilly & Associates, and is currently working on an update to that book. His software can be found at ftp://ftp.linux.it/pub/People/rubini/.

Alessandro is a strong supporter of GNU and the Free Software Foundation. He calls Linux “GNU/Linux” in the speeches he gives advocating Linux. People tell him he sounds and looks more like Richard Stallman every year. He told me, “If a package is not open source, its author doesn't deserve any help from unpaid contributors.” In fact, he believes the biggest threat to Linux is proprietarization: “the rise of products that are based on free software but are not free as a whole. There's a risk of forking the user base and repeating the UNIX fragmentation.”

Alessandro has a home page at http://www.linux.it/~rubini/ and can be reached via e-mail at rubini@linux.it.

Tommy Thorn

Tommy Thorn was a computer science student at DAIMI, University of Aarhus in Denmark, when he saw Linus' first announcement about Linux. At version 0.01, he felt Linux was “promising”, and at 0.11, quite usable. But to run it himself, Tommy needed support for his SCSI adapter, Adaptec 1542b, so he wrote the initial driver for it—and was soon in the Linux business.

When asked what Linux needed in order to succeed, and its biggest threat, Tommy waxed eloquent. Here is his answer verbatim:

> Intellectual property (i.e., patents) is the greatest threat to free (speech) software and Linux in particular. In fact, I believe it is a threat to humanity.
>
> As a UNIX, Linux is already very good and steadily improving; however, the UNIX model has it share of problems. For example, the anarchical, disintegrated and inconsistent tradition of UNIX configuration, and the fact that setting up devices requires a lot of low-level knowledge and much manual intervention. Better tools can patch the problem, but only a radical change will really improve the situation. However, the thing about UNIX (and thus Linux) that bothers me most is

> the security model, especially in these Internet days of heavily interconnected machines. The three-level permission model is grossly naïve. Is it too much to ask to be able to share permissions with a selected set of users, to specify permissions on [an] object with common properties instead of each individual file, to be able to launch subprocesses with restricted rights, etc.? Other operating systems such as HURD, L4, VSTa and EROS incorporate interesting solutions, and there has been work on adding capabilities to Linux that holds some promise, but we need some big changes, changes that I fear are too drastic to ever be accepted into Linux. The first step is to agree upon the problem.
>
> For the desktop, user interface is really important. Gnome and KDE have done much to make UNIX easier to use, but the underlying model is still one of the flat file and the hierarchical file system, which isn't necessarily what is best for users (Anti-Mac, useit.com). In consequence, applications invent their own, but mutually incompatible, richer model, such as the mailbox formats, configuration file formats, etc. The result of this is the lowest common denominator is the ASCII file with no semantic content. Maybe XML will finally provide a path to richer files that will allow applications to share data in a more intelligent manner, but other issues remain, such as object replication, migration, object and process persistence, cross reference, etc., that all must be implemented on top. The challenge is to achieve this in a consistent manner across all applications.
>
> The area most likely to see innovation in user interface is the embedded applications. This is a new domain for Linux, and it calls for new solutions. As an example of the willingness to part with tradition, part of the community has abandoned the X Window System in favour of Microwindows, a new and smaller windowing system written from scratch for embedded applications.

Tommy Thorn can be reached via e-mail at thorn@brics.dk.

Jon Tombs

Today, Jon Tombs is a professor at the University of Seville's Engineering School in Spain and finds “little time for Linux.” But in the beginning, he was involved mainly in X issues. He relates it this way:

> I worked on the CD-ROM code (Mitsumi driver), as I had an unsupported Mitsumi single-speed CD-ROM. I worked on bits of the NFS code, as I wanted to use Linux on the university network. I worked on the Colorado FC10 floppy tape interface, because I had one. I worked on kernel modules, as they allowed me to work on the CD-ROM and FC10 drivers without rebooting. I was also interested in accelerating graphics and the X Window System (partly for xpilot and xblast gaming) and started compiling the X386E Xserver for Linux. I was quite proud of the fact that for a time, Linus was downloading my Xserver compile.
>
> I soon got involved with the XServer S3 support (together with Amancio Hasty of the NetBSD band). He had good contacts within S3 Inc., and during a conference in Denver, I managed to upgrade my original S3 card for the latest unsupported card (S3 801).
>
> This led to my involvement with the founding of the XFree86 Project, Inc., of which I am still a founding board member. I still write applications from time to time, but my work schedule doesn't allow me time to modify the kernel, unless I touch a bug that annoys me. I still give support to students who work on Linux or applications for Linux. EtherApe is a recent application by a student of mine.

He likes Linux best for its productive environment, saying, “Windows makes me feel [as if] I am computing with one arm tied behind my back.” He admits that knowing UNIX before MS-DOS *may* have warped his expectation, that a computer should be “a tool, not an application.”

When asked if Linux should have stayed within the hacker community, he told me:

> I would have preferred that the non-hacker community had stayed more disconnected from the hacker core. Many hackers have been molested and insulted by the e-mails of “dumb users.” On more than one occasion, this has prompted developers to give up. The “signal to noise” feedback a developer receives nowadays is much worse than it was. Hopefully, the distribution vendors will in the future be the firewall between developers and irate users.

Jon Tombs can be reached via e-mail at jon@gtex10.us.es.

Theodore Ts'o

Theodore Ts'o is another person who didn't get back to us, but whom we know enough about to write something anyway. Ted is the author and maintainer of the **e2fsprogs** package, which contains the EXT2 file-system utilities. He is one of the core Linux kernel developers and serves on the technical board of Linux International. He currently works for VA Linux Systems, but was a long-time systems programmer for MIT before joining VA. He is much sought after for his knowledge and lecturing abilities.

While at MIT, he headed the Kerberos development team, was a member of the Internet Engineering Task Force and worked on the Telnet Encryption Specifications. He took his degree in computer science in 1990.

He is a fan of Gilbert and Sullivan and enjoys folk dancing, ham radio, cooking and bicycling. More about Ted can be found at web.mit.edu/tytso/www/home.html, a site which includes a picture of him proving that "playing with a Van de Graph generator can be a shocking experience."

Other credits include authoring the serial driver, e2fsck, job control and system call restart code, ramdisk device driver and loopback device driver.

Fred van Kempen

Fred admits he has "made a lot of money off the Linux thing and cannot say it feels bad." He is perhaps best known for selling the Linux domain names—both Linux.com and Linux.net—although he "cannot share any details regarding those deals." He is also "a medium-to-large shareholder of some U.S. (Linux-related) companies." There are critics who blast Fred for profiting from Linux, but he has contributed heavily to the development of Linux. The money issues are his business.

Fred thinks Linux belongs in the "Big Bucks World ... where decisions about future technologies are being made. Linux is *there* now, and thus will be part of that decision making." Adding money to Linux development is a good thing, because developers "can now be paid a decent salary to work on Linux stuff." It's hard to argue with people enjoying their work.

UNIX and Minix were Fred's first programming interests. He relates,

> I was a UNIX (2.9BSD, 2.11BSD, 4.2BSD and V7) systems programmer already, playing with DEC PDP-11's at home. I played a major role in the development of MINIX, and Linux was the most natural next phase.

He was introduced to Linux by a fellow MINIX developer, Miquel van Smoorenburg, who urged him to try it. He did, and “the rest is history,” Fred says, smiling.

His primary areas of focus on Linux were networking and the high-performance/high-availability part. This work has continued, as Fred works as a senior consultant for Nobel Van Dijk & Partners (the Netherlands). He works on “big iron machines for enterprise-level customers ... which means very large networks, massive server parks and lots of Windows NT and Tru64 UNIX on FAT Compaq servers.” He holds most major certifications, and is “working” toward his RHCE!

He would like to see Linux with more “FAT server support”. He wants Linux to “scream” on big iron, believing it “can actually replace several *other* UNIX systems and/or Windows NT.”

As for the desktop? “The desktop means only one thing: business applications. For many reasons, the business desktop means supporting Microsoft *applications*.” Fred thinks that Linux, if it wants to be used for these things, “will have to at least resemble that desktop.”

Outside work, Fred spends some days “living the good life in sunny Southern California with my fiancée. Yes, there is life outside of these things [computers]. Uhh ... I think. I've been told stories about beaches, the ranch and kids.”

Fred van Kempen can be reached via e-mail at fred@nobel.nl.

Patrick Volkerding

Patrick put out one of the first distributions of Linux—Slackware. For a long time, Slackware was the most-used distribution available: if you were running Linux, odds were good you were running Slackware. Like many others, Patrick became involved with Linux as a university student studying UNIX, because it was free and was compatible with his machine—386BSD was not.

In those early days, Linux had a lot of bugs, and Patrick found himself fixing a good many of them in his version—love having that source. When he read Internet postings from others who were having the same problems, he decided to make his changes available to others on-line. Walnut Creek stepped in about this time, and offered him archive space at their site. Eventually he went to work for them, and they now sell the Slackware distribution.

Patrick feels strongly about the Open Source movement, but still thinks people should support whichever product is best, whether it is free or commercial. He believes that in order for Linux to continue to expand, it will need the support of both sides. He also feels that making Linux easier for new people could result in making it more complex, and he has no intention of “dumbing down” Slackware only for it to end up on the desktops of newbies who don't care how the system works, only that it does. He says, “I appreciate the simple elegance of UNIX. [Dumbing it down] would spoil its flexibility.”

When asked who he felt was the most influential person, other than Linus, in the community today, Patrick replied,

> I'd have to say Jon “maddog” Hall of Linux International, because at a time when Linux seemed to be viewed as a hacker product that couldn't really be trusted and didn't have any centralization, he put a friendly and trustable face on the OS and seemed to say all the right things to get the mainstream computer world to start taking Linux seriously. He's also a master at fostering cooperation and compromise among all the individual groups working on Linux projects. I'm really glad we have someone so level-headed representing the Linux community to the mainstream.

Patrick can be reached at volkerdi@slackware.com, and Slackware's home site is http://www.slackware.com/.

Matt Welsh

Matt Welsh is a very personable young man, who still looks 18—maybe because he's not much older than that. Matt is a very busy man who enjoys traveling. He visited Nepal last December, and you can see pictures of this trip on his web page at http://www.cs.berkeley.edu/~mdw/. He is an avid music lover who plays acoustic guitar and loves jazz.

Currently, he is a Ph.D. student at UC Berkeley, where he just received his master's in December 1999. He is doing research there involving “scalable Internet service architectures with a focus on implementing high-performance systems in Java. I am affiliated with the Berkeley Ninja and Millennium projects.”

Matt is best known as the author of *Linux Installation and Getting Started* and *Running Linux*. He has also been the Linux Documentation Project coordinator, the HOWTO coordinator and writer, the maintainer of sunsite.unc.edu Linux documentation archives and the moderator of comp.os.linux.announce.

We weren't able to get in touch with Matt. But since he has written for us in the past and we hope he will find time to do so again in the future, we wrote this for him. Most of it is true. :)

Lars Wirzenius

Lars Wirzenius (such a fluid, melodic name) is Linus' friend and was also a student at the University of Helsinki at the same time Linus was. He remembers,

> Linus showed me a program that had two threads that wrote As and Bs, respectively, to the screen. That was the beginning; it evolved into something more interesting later.

If you are interested, more stories from the past nine years can be found at www.iki.fi/liw/texts/index.html#linux-anecdotes.

Most people know Lars through his efforts for the Linux Documentation Project and as moderator for the comp.os.linux.announce newsgroup. The only code he wrote that went into the kernel was “the part of the printk routine which prints out messages to the console, more specifically, sprintf.” Most of his programming efforts have gone into applications, not the kernel.

These days, he works for WapIT (http://www.wapit.com/) writing free software —everyone's dream job, right? Currently, his work focuses on the “WAP and SMS gateway called Kannel (http://www.kannel.org/).” While he is still involved with the Debian project, maintaining packages, he does not have as much time for this as he had in the past. To counteract the stress of the job, he likes to “relax by having fun with friends, reading, watching movies and playing role-playing games.”

Lars is a very quotable guy. For example, consider the following statements he made to us:

- I think the current popularity of Linux is very nice indeed.
- Commercialization is good, as long as co-operation continues.
- Proprietary file formats or protocols are really, really bad.

- Software freedom is important, even though I don't require all software to be free.
- Since I've been having lots of fun programming, I can't ever feel that time has been wasted.

Lars can be reached at liw@iki.fi.

Rogier Wolff

Rogier Wolff owns his own company, BitWizard, just so he can work full-time on Linux. He is now in a position to reject work that doesn't relate to Linux. He writes commercial Linux device drivers and urges you to write him if you need a driver for a special device.

When he first began working with Linux, he contributed fixes and worked on the “memory management system.” He liked the fact that his garbage-collecting code to fix memory fragmentation would be incorporated into the Linux code base, unlike Minix development. He has enjoyed working with others over the Internet and finds it particularly rewarding to work with Alan Cox, as “Alan is better [than Linus] at giving short feedback about what's wrong with the suggestion, allowing me to improve whatever I'm doing.”

About the future of Linux, Rogier told us,

> For the future, Linux needs to be able to adapt well to different situations. On an end-user workstation, wasting, say, 1MB in unnecessary drivers is a convenience issue that is well worth the RAM. It makes life easier for the end user, and RAM doesn't cost that much. On an embedded application, Linux still needs to be able to be configured for the minimal requirements that apply there. On an enterprise server, you need to be able to configure Linux to do well on a machine with lots of RAM and CPUs.

When asked how he felt about commercial software, he replied,

> Good—an operating system should provide a platform. The free software idea is that when it's easy to make/expand stuff, then the programs can be made available for free. However, there will always be areas where significant “boring” programming is required to make something work. People will have to pay for that development. Companies should be allowed to chose whatever system they think is appropriate for getting a revenue stream from their products.

He also feels that the reason Linux works so well is because all code must be approved by a maintainer—Linus. This type of license is good for other types of software as well, keeping the patches from becoming incompatible.

Rogier Wolff can be reached at r.e.wolff@bitwizard.nl.

Eric Youngdale

Eric became involved with Linux before the 1.0 kernel release, and very much enjoyed the camaraderie among the various developers. He found the Linux community to be friendly and supportive of newcomers, as opposed to the BSD community that had a “tendency to argue, flame, and in general, eat their own young”.

During the day, Eric was a research scientist at the Naval Research Lab in Washington, DC. At night, he worked on the kernel, adding support for core files. Doing this work convinced him that the a.out file format needed to be replaced, so he began working on kernel support for ELF. He also did a lot of work on the linker, assembler and dynamic loader. Later, he worked on writing an iso9660 file system to handle CD-ROMs and on the SCSI subsystem. These days, the SCSI subsystem is the only place he “still sticks [his] fingers.”

When asked if there was a world outside computers, he replied:

> Absolutely, I have to frankly admit there are times when I really resent the amount of time I spend on free software, and these are the times when I completely stop reading e-mail. In order to make the most of what limited time I have, I am trying to be fairly selective about what I get involved with, and really only work on things where there isn't anyone else taking a leadership role. There are also times when things get quite intense at work [so] that I don't have the interest or the energy to sit down in front of my machine when I get home.

His outside interests are many and varied, from roller hockey to classes on reading, writing and speaking Chinese. He likes to scuba dive and ski when he gets the chance and spend time near the ocean in the summer.

Eric feels Linux wouldn't have been possible without the GNU tools and that we owe the free software community a “huge debt of gratitude, but calling it GNU/Linux seems silly.” Regarding threats to Linux, he sees two: Windows NT and fragmentation. About fragmentation, he said:

> The danger I see for Linux is all of the different distributions that are appearing—there have to be close to six different boxes down at CompUSA these

> days, and we are already starting to see subtle differences in the way these things work (KDE/GNOME is an example, here). I see no good that can come out of having all of these different choices.

Eric has many strong views and is not afraid to say things that might not be agreed with by the majority—an excellent trait in one of the leaders of our community. He can be reached by e-mail at eric@andante.org.

Credits

email: eric@andante.org

Archive Index Issue Table of Contents

Advanced search

My Life and Free Software

Jon “maddog” Hall

Issue #74, June 2000

The godfather of Linus' children and the glue that holds the Linux community together tells us a bit about his life and involvement in the programming community.

Often I get questions of how I got started in Linux, why I like Linux, or why I spend so much time on Linux. In this article, I hope to show that the concepts of free software and the gift of intellectual property to complete strangers are not necessarily new ideas.

The year was 1968. I was a college student studying electrical engineering at Drexel Institute of Technology (now Drexel University) in Philadelphia, PA. As a cooperative education student, I was expected to go into industry for a six-month work period. With great luck, I managed to land a job at the Western Electric Plant in Baltimore, Maryland.

The plant, which employed thousands of people, principally made wire for the Bell System. As part of the engineering department, they had an IBM 1130 computer that was run by a small staff of programmers/operators. The 1130 was a single-task machine, where you would typically write a program in FORTRAN, punch it on cards, and compile, link and run your program all at one time. Since the 1130 also had console lights and a very noisy disk coil, you could “see” and “hear” your program run. With a little practice, you could even tell when your program was in trouble by watching the console lights and listening to the disk drive “grunt”.

Back in those days, people did not have computers in their homes, and even the most forward-thinking high schools usually did not have a computer, so there was no way I could have learned to program an IBM 1130 in school. So I jumped at the chance to participate in a correspondence course called “Programming the IBM 1130 in FORTRAN”, sponsored by Western Electric. Over that summer, I learned to program the machine, completing a few small tasks

for my supervisor, John Kammer. The people who ran the computer in the lab were very helpful to me, answering my many questions very patiently, which made a lasting impression.

Drexel had two large machines: one was a Burroughs B5500, the other an IBM. The IBM was shared as a facility with the University of Pennsylvania and Temple University. Both the IBM and the Burroughs were programmed with punched cards, and neither were available for me to “watch the lights and hear the grunts”. In other words, they were “glass house” machines. Discouraged by this, I looked around the school and found an electrical engineering lab with a couple of PDP-8 “minicomputers”, a Linc-8 minicomputer (all made by Digital Equipment Corporation) and a Data General Nova. These were “hands-on” machines, using an ASR-33 Teletype (five characters per second output) as printer/console terminals and paper tape as the load medium. Loading the editor took five minutes; loading the assembler took 15 minutes. A complete round of “edit, assemble, load program and run” took about half an hour, even with no editing time taken into account. A person got used to checking their code very carefully, in order to find as many errors on each run as possible.

Drexel offered no courses in computer languages. If you wanted to learn FORTRAN, you learned it on your own. If you wanted to learn assembly, you learned that on your own. Fortunately, I somehow made the acquaintance of the Digital Equipment Corporation salesman who gave me a couple of paperback books, one of which was on the fundamentals of programming in assembler, the other entitled *Programming the PDP-8*. From these two books, I started the task of learning to program the machines in assembly language. While learning assembler language programming from a book might sound daunting to some, the books (part of a handbook series put out by Digital) were very good, and along with the other students using the lab (the “lab rats”), we managed to coach each other through the process.

Later, several of the “lab rats” started a computer club. Our goal was to try to teach others how to use the minicomputers. We had some help from two sources: the DEC salesman (who gave us more books and encouragement) and the organization known as DECUS. Even in those early days, DEC's User Society had student chapters, and they also had a catalog of freely distributable software—free for the copying cost of transfer to paper tape, or (for the “really wealthy”) magnetic tape. There were hundreds of titles, from text editors to programs for doing mathematical analysis of all types. Each of these programs had been written by a DECUS member and submitted to the DECUS library for free dissemination.

I never forgot the kindness of that Digital salesman, nor the philosophy of the DECUS organization. It would have a profound effect on my later life, both as I chose new jobs and as I dealt with other people.

After graduation, I started looking for a job. I had several offers from different companies including the Bell System, but they all wanted people to program in high-level languages, and I wanted to continue to program in assembly. My chance came with Aetna Life and Casualty, an insurance company in Hartford, CT.

At the time, Aetna was the "largest multi-line insurance company in the free world", which goes to show that if you use enough qualifiers, you can be the best of anything. Nevertheless, Aetna employed over 4,000 programmers, operators and computer operations people at their headquarters in Hartford, and they developed some of the first real-time transaction systems in the industry. They had a massive floor of IBM equipment, and they were the "largest commercial customer of IBM in the free world". I was hired to write programs in IBM assembly. I told the interviewers I had never coded in IBM assembler, but they said they felt I "could learn". So before I left for Hartford, I went to a bookstore and purchased a book on IBM assembly language and read it. Between that book and the IBM *Principles of Operations of the IBM 360* manual, I learned the language and architecture.

While working at Aetna, two major things occurred. First, I started going to graduate school at night for my MSCS; the second was making friends with some of the operators of the machines. The second event brought me out of a shell I was in, and made me more outgoing. It also gave me physical access to the computer floor, something very useful from time to time as a programmer at Aetna. I also noticed that my tapes got mounted faster, my listings were sorted nicer, and various good things happened to me because I liked these people and treated them with respect. Other programmers who did not treat them with respect often had a hard time getting good service. Another major lesson learned in life.

After receiving my MSCS and having worked in two major divisions of Aetna over a four-year period, I decided to try my hand at teaching. I answered an advertisement for an assistant professor at Hartford State Technical College, a small two-year technical college in Hartford. Before taking the job or even applying for it, I spent a morning at the school looking at the lab, browsing through the college catalog and talking to the students.

I liked the lab at Hartford, because (unlike many schools of the day) it was not keypunch-card-oriented. Instead, they had twelve terminals hooked up to a PDP-11/70 timesharing machine, along with a stand-alone PDP-11/34 with a

graphics tube and a large Gerber plotter for making drawings. The curriculum taught courses such as operating system design and compiler theory, both of which I felt were necessary if a programmer was truly to know what they were doing. Finally, the students were friendly and seemed to be dedicated to learning computer science. Remember, this was 1977, and for most of these students, HSTC was the first time they had set hand to computer keyboard.

I gave Aetna a two-week notice and took a $3,000 pay cut to teach at HSTC (to put things in perspective, in 1975 I bought a brand-new Toyota pickup truck for $5,000). But for the next four years, I enjoyed teaching at the school.

Most of you have heard that my nickname “maddog” came from the school and from my “discussions” with the Dean of Instruction. Even with these “discussions” (and sometimes even because of them), I look back on HSTC with fondness. Taking students who have no knowledge of computers at all and training them to be good programmers and designers, so they could enter the work force and get good jobs, has to be one of the most rewarding feelings in the world, no matter what the pay scale. Unfortunately, I have a rather eclectic lifestyle, and after four years, it started to catch up with me. Sadly, I left HSTC (after giving a year's notice) and went on to the next phase of my career, Bell Laboratories, to the dismay of the students, but to the cheering crowds of my creditors.

When I started interviewing at Bell Labs, the management wanted me to be the system administrator for a CDC Cyber machine. The CDC was a fascinating machine, and this would have been a good job, but I had heard Bell had their own operating system called Eunuchs, and I wanted to learn that. Eventually, the Lab management agreed, and I was hired as a system administrator for an operating system I had never seen.

The first few months of being a UNIX system administrator were “difficult”. Learning this new operating system that had no real good documentation, seemed to have programs scattered throughout the file system, and even had a deep hierarchical file system (instead of a flat “catalog” file system) was hard for me. Fortunately, I met two senior engineers there who (although they had not been trained in computer science) were administering two Digital machines, a PDP-11/70 and a VAX 11/780. They gave me guidance until I was ready to take over responsibility. However, after a short time I “got the swing of it”, and I also started teaching at night at Merrimack College in North Andover, MA. I did both of these jobs for four more years.

While at Bell Labs, I met a Digital salesman who told me about a project at Digital to create a commercial version of UNIX for the PDP-11 and VAX line. Since I was having some philosophical differences with the management at Bell

Labs, I decided to leave them and go to Digital. There, I met a group of young, enthusiastic engineers who were determined to make the world's best UNIX system. It was too bad that their management did not share that vision. For the next sixteen years, we struggled to create and sell a UNIX product inside a corporation that (for the most part) only wanted to say “VMS” (later “OpenVMS”) and “Windows NT”. We struggled (and finally won) the battle of getting GNU software onto our distribution, after almost every other vendor was bundling it. It was at the start of these fifteen years that I was (once again) thrust back into DECUS and started attending USENIX conferences.

USENIX in 1983 was an exciting time. People came to this technical conference to talk about how they were improving the UNIX operating system. While many talks came from academia, some were from companies like Mitre, BBN and others that needed to have UNIX move forward for their own business reasons. After a talk was given, the UUCP address of where to get the code would be listed, or else a complete source code listing was included in the proceedings. However, as more and more companies got into the market of selling UNIX distributions in binary form, and more and more companies started buying UNIX this way, the availability of source code almost completely dried up. Although USENIX kept providing the venue for concept and knowledge exchange, it took many years and the insertion of the Open Source community before the same feeling of excitement came back.

While at Digital, I was a software engineer, a product manager and finally, a technical marketing manager. As a technical marketing manager, I had three main jobs. The first was to take complex technical concepts and present them in simple terms to customers; the second to take customer's requirements and feed them to engineering; and finally, to help create marketing opportunities for Digital. This included working with DECUS to make technical presentations relevant to the customer's needs.

As part of DECUS, there was a special interest group (SIG) called UNISIG which looked into “everything UNIX”. Kurt Reisler, the group's chairperson, had heard of a small operating system that had been started in Finland and was now being developed all over the world. He wanted to invite the architect of this operating system to DECUS in New Orleans in May of 1994. After watching Kurt send many, many letters to potential funders of the ticket, I decided this would be worthwhile for our customers and convinced my management to fund the ticket and hotel costs. It was in New Orleans that I first saw Linux and met Linus Torvalds.

After DECUS was over, I took Linus out on the Natchez, one of the riverboats that goes up and down the Mississippi River. We had dinner and discussed Linux. I asked Linus if he had ever considered porting Linux to a 64-bit

processor and a RISC chip, to make sure it was portable. Linus told me he had considered that, but was having trouble getting an Alpha system from the Digital people in Helsinki. Knowing Digital's procedures, I could imagine the Helsinki people were trying, but having difficulty. I went back to my office, called a few friends, pulled in a few favors, and an Alpha system was on its way to Linus' apartment.

As I was getting the system for Linus, I became aware of how many engineers inside Digital were already using Linux, and that a group in Digital Semiconductor was trying to port Linux to the Alpha, but as a 32-bit port, not a 64-bit port as Linus was doing. With a little convincing, I got them to join forces with Linus.

In January of 1995, the project actually got underway, and as it unfolded, I found many people in the Linux community who wanted to help. Some people who already had access to an Alpha immediately started working on the code. Others actually bought their own Alpha to help with the project. In November of 1995, the first distribution of Alpha Linux came out. I was proud of the fact that the Linux community thought so much of Alpha, Digital and their own Linux system that they had all cooperated on this project.

About the same time, my travel schedule picked up and I started going all over the world to speak on Digital UNIX (the commercial system available from Digital). As I went from customer to customer, I would ask the same question: “Do you use Linux here?” Almost every one had the same pattern—the management said “No”, and the technical people would say “Yes, but don't tell our managers.” As I continued on my trips, more and more of my presentations became oriented toward Linux, and I began to write articles for both internal use and *Linux Journal*.

In 1996, the original Executive Director for Linux International stepped down, and I was elected to this position. For three years, I volunteered as Executive of Linux International while doing my duties at Digital Equipment. At first, it was relatively easy to work for both, but as the Linux marketplace heated up, I put in more hours on both sides. Finally, I had an offer from Larry Augustin of VA Linux Systems to have VA Linux pay my salary but work for Linux International full-time, so I left Digital and have been working on Linux International full-time since the spring of 1999.

I have been in the industry more than thirty years. All along the way, there have been people helping me out, giving me guidance and advice. While some of it was bad advice, much of it was good, and most of it was given with the best possible intentions. I have participated in the industry in both profit-making and non-profit-making projects. I have tried (not always completely

successfully) to keep my word at all times, and to “do right” by the “customer”, whoever I considered “the customer” to be.

I have found in the Linux community what I feel is the best of all of my experiences. Good technical people to work with, open exchange of ideas, excitement and enthusiasm for what is going on. Young people (no matter what physical age) who strive to do their best, but also know how to have fun. The warmth of meeting new friends, and in some cases even being welcomed into their family. The potential of Linux in education, embedded systems, supercomputing, as well as the “normal” desktop and server situations is both to save money and extend computer science.

I know that in some parts of the Linux community, there is the feeling that the “suits are taking over”. From my experiences, labeling everyone in business a “suit” is as much an act of bigotry (and with all the same bad features) as any other type of bigotry. Many people I have met in industry who are moving into the Linux marketplace *do* want to learn what the Linux community is about, and they *do* appreciate the culture as they try to understand it. I have faith that the Linux community can expand to embrace these commercial interests without diluting our message, but we have to learn that it is easier to extend the hand of friendship rather than the battle axe (unless the battle axe is really needed).

The picture on this month's cover is very telling. And while some of my jobs over the years have required me to wear a suit on occasion, I am much happier as a shorts-and-T-shirt guy. While I have held jobs in commerce most of my life, I was and am happier when the free exchange of ideas (including source code) is available. And while I am definitely an American, I have been fortunate enough to meet and enjoy many different cultures and peoples. I hope Linux will be one of their main paths to computer autonomy, and for some emerging nations, development of their own computer industries.

Finally, over a thirty-year period I have been fortunate enough to meet and know some really fantastic pioneers of the computer industry, some famous and some not so famous. In the Linux community, I have already met hundreds more. As the next millennium moves on, I look forward to meeting thousands more both over the Internet and face to face.

Jon “maddog” Hall (maddog@valinux.com) is Executive Director of Linux International and Director of Linux Evangelism at VA Linux Systems.

Archive Index Issue Table of Contents

Advanced search

Monitor diald from Anywhere on Your LAN

Ed Beroset

Issue #74, June 2000

Find out what's happening when using IP Masquerade and diald to access the Internet remotely.

Imagine making a telephone call on a system with no dial tone, no ringing signal and no busy signal. You'd just dial your number and wait until someone either answered or did not. It might be a frustrating experience after a while. This is similar to the experience of someone on a LAN using **diald** on a remote computer without any way to monitor the status of the remote link, and the problem I set out to solve about a month ago.

My home LAN includes my main Linux box and a number of other computers running some combination of Linux and Windows 95. With the main Linux box configured to use diald and IP masquerade, my wife can use her Win95 computer to check her e-mail, surf the Web, and do other Internet tasks without having to do anything with the modem, dial strings or the Linux box. She simply starts Netscape, goes to a site out on the Web, and waits. The Linux box dials the ISP, authenticates automatically and establishes the connection in about thirty seconds—unless the modem gets a busy signal. Unfortunately, there wasn't any feedback mechanism to tell the Win95 box about the state of the connection. This article describes my solution to that problem.

Using Linux to Connect a LAN to the Internet

A single Linux box, equipped with a modem and configured to use IP masquerade and diald (which is short for “dial on demand”), allows every computer on the LAN to “think” it's directly attached to the Internet. An Internet request originating on a node of the LAN gets sent to the Linux box. If diald determines that the serial link to the Internet service provider (ISP) is not currently active, it initiates the dial sequence (hence “dial on demand”). When the connection is made, the Linux box authenticates with the ISP's computer, and the link to the Internet is established. IP masquerade allows that single

connection to be used by every computer on the LAN by examining each packet destined for (or coming from) the outside world and substituting addresses where appropriate. The ISP side of the connection sees traffic only to (or from) the single IP address with which it authenticated—addresses in reply packets are then substituted back into place before the IP Masquerade box places the packets on the LAN side. While very complex, this all works remarkably well when properly configured.

When my wife checks her e-mail, however, she has no way of knowing the current state of the link. Her computer attempts to fetch the mail, sending a request to the Linux box running diald. The Linux box initiates the dial and authentication with the ISP, but this takes about 30 seconds, and the Windows 95 machine isn't that patient. When I use her machine, I just use TELNET to get to the Linux box and observe diald's progress by doing something like **tail -f /var/log/messages**, but this requires access to the system logs and some knowledge of how to interpret them.

An Easier Way from the Bottom Up

The primary goal of the system was to provide an easier way to monitor the status of the remote link. Since I already had an HTTP server, Apache, installed and running on my Linux box and all of the computers in the house have some kind of web browser, it seemed an HTTP-based solution would be simplest. It has the advantage of requiring changes only to the Linux machine, making deployment quite simple. Conceptually, what's needed is:

- a way to monitor the state of diald on the Linux box
- a means by which a cgi-script can get that information and pass it to the client
- some means of automatically updating the information on the client side

Starting from the bottom and working up, you'll need to get **chat** working first. I use a chat script, /etc/ppp/chat.mindspring (see Listing 1) to dial in to my ISP, Mindspring. I found both the ISP HOWTO (metalab.unc.edu/mdw/HOWTO/ISP-Hookup-HOWTO.html) and the diald mini-howto (metalab.unc.edu/linux/HOWTO/mini/Diald.html) very helpful in getting chat and diald working well. The IP Masquerade mini-howto (metalab.unc.edu/linux/HOWTO/mini/IP-Masquerade.html) was instrumental in helping me configure IP Masquerade. By reading carefully, planning meticulously and working methodically, you should be able to get your system working without too much difficulty within a couple of hours.

Listing 1

This chat script uses the **SAY** option, which allows the printing of strings from within chat to STDOUT. However, STDOUT must be redirected to a file to be useful in this application. This is done by using a Perl script to run chat (see Listing 2).

Listing 2

This script is necessary, because we use the single diald FIFO (first-in, first-out) to send back messages. As you can see from the script, we use the "message" prefix to send messages to the diald control FIFO. Additionally, the complex method by which we call chat is required, because old versions of diald call the connect script with STDERR directed to the output device. Normally this works just fine, but we need to send the STDERR results to a file and not to the modem. Versions of diald from 0.98 on will direct STDERR output to monitors by default, making this process much simpler. Fortunately, diald sets a number of environment variables which are useful for this script, including **MODEM** and **FIFO** which allow us to write a fairly portable script.

Listing 3

In the diald control script (Listing 3), note the connect line (line 3) which runs the **runchat.pl** Perl script. It has a single argument, which is the name of the chat options file used. This is useful in the event that multiple diald dæmons are running simultaneously. In my case, one diald establishes a connection to my ISP. Another diald dials into my office LAN. The only required difference between the two diald scripts is the argument provided to the runchat script. Other differences in my case are the pppd-options file specified in the last line, and the **defaultroute** line (my office connection uses a separate routing script).

The diald control script will need to be changed somewhat if you're using a more recent version of diald. I'm still using diald version 0.16.4 and the 2.0.36 kernel, but I plan to upgrade to the latest diald, version 0.99.1, as soon as I move to the 2.2 kernel. In addition to a number of bug fixes, the new version includes features which would probably be useful for a script like this. You can find out about the latest version of diald, now maintained by Mike Jagdis, at http://diald.unix.ch/.

A few more pieces are required to make all of this work. One is the script that establishes a connection to diald and listens via monitor FIFO to whatever diald and chat have to say. This script is reproduced in Listing 4 and is a Perl implementation of a poor man's dæmon.

Listing 4

This simple script maintains a file (/tmp/diald.status) which contains the last few relevant entries. Following the lead of the Tk **dctrl** program, the file is cleared whenever the **CONNECT** string is received. This is a useful way to slice things, since the user is likely interested in only the current connection progress and not in historical data. The **CONNECT** message is the first one transmitted via FIFO when the need for a new connection is determined by diald. Note also that the messages are detected and passed unparsed to the output file.

Finally, the end user has to actually see the results of all this work. This is done by having a special web page on the Linux box, which causes the client browser to spawn a separate status window, allowing the user to monitor the state of the Internet connection while browsing or downloading e-mail. The browser's home page is set to an appropriate local page (i.e., a page served by the Linux server) containing the following JavaScript tag:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- comment hides the script from old browsers
newWindow = window.open (
 'http://clyde/cgi-bin/dialdstate.pl?', 'dialdstate',<\n>
 'scrollbars=no,status=no,width=365,height=285, status,
marginheight=2, marginwidth=2');
// end of hidden portion. -->
</SCRIPT>
```

The height and width were chosen empirically, and “clyde” is the name of the server box as presented to the LAN. All that remains is the content of the **dialdstate.pl** program, which is in Listing 5.

Listing 5

This script does very little except set up a simple page with JavaScript. The only reason it is not just an ordinary HTML page is that we use an automatic reload to cause the browser to refresh the page automatically every 7 seconds. This is the **setTimeout("location.reload() ", 7000);** line, which executes the **location.reload** command every 7000 milliseconds. This causes the Perl script on the server to rerun and provide the latest data to the user. This is fine on my home LAN with few users and little traffic, but you should carefully consider the load implications before deploying this on a larger LAN.

Future Enhancements

This solution meets the goal of providing link status information to the user, with minimal effort required by the user and no custom software on the client side. As with any project, though, there is room for continued improvement.

Possible future enhancements include upgrading to the latest version of diald and exploiting more of its features, such as reporting on the state of multiple

connections. Another improvement might be to rewrite the “dæmon” in C++ rather than Perl. The existing version works, but it's not particularly elegant.

Also possible is the addition of remote link control via JavaScript buttons (e.g., Up, Down, Block, etc.), but this should be done carefully to avoid opening a security hole. Allowing external sites to see the status of the LAN's dialup link wouldn't necessarily be a security problem, but allowing external sites to control it would be. On my Linux box, I have IP masquerade set up such that the HTTP server is visible only from within the LAN. An alternative approach might be to use the security methods in Apache to limit access to sensitive web pages.

Conclusion

IP masquerade and diald are useful tools for allowing Internet access to an entire LAN, although sometimes it's nice to know what's going on remotely. By using the scripts presented here, even novice users on your LAN can enjoy some of the power of Linux without even knowing it's there.

email: beroset@mindspring.com

Ed Beroset (beroset@mindspring.com) works as a firmware manager at ABB Automation in Raleigh, NC. On the long commute to and from his Chapel Hill home, Ed often engages in another of his hobbies, amateur radio, as KF4UQX. His wife Marilyn is a psychotherapist and part-time software muse.

Archive Index Issue Table of Contents

Advanced search

wxPython, a GUI Toolkit

Hugues Talbot

Issue #74, June 2000

You'll feel much better after reading about this new cross-platform toolkit written in Python and wrapped around wxWindows.

To many people in the “real world”, MS Windows is an inescapable, however unfortunate, fact of life. Some of these people might even have clients or bosses who require a “standard” Windows GUI application at the end of the day. For many developers, being able to choose the best tool for the task at hand, rather than being told which one to use, is an essential element of their quality of life, and Linux is a pleasant development environment.

It is in fact now possible to do quality, rapid GUI development for Windows on Linux, and is becoming increasingly possible for other GUI systems as well, such as Apple Macintosh, BeOS or OS/2. One of these tools, which I present in this article, is wxPython, which is not only based upon a proven toolkit, wxWindows, but uses our favorite language, Python.

Portable GUI Toolkits

There is no lack of choice of GUI development systems or toolkits under Linux, and quite a few of them can be called portable, i.e., work on at least two different platforms, for example, Linux/UNIX+X11 and Windows. Many of them are free, but not all of them are in usable form (i.e., documented, maintained, relatively bug-free and feature-rich). However, we are fortunate that such tools do exist. In fact, a well-maintained web page, http://www.free-soft.org/guitool/, lists an impressive number of such toolkits. The problem is deciding which one to choose, since life is too short to try them all.

How to Choose?

Life is a series of compromises. Often, the problem is finding out which set of drawbacks one is prepared to live with in return for which benefits. For one of my projects, I finally settled on the following list:

- compatible with Linux, Windows and Python
- usable
- large enough widget set
- not too slow
- compact
- easy to learn
- converts to C/C++ easily after prototyping stage, if necessary
- looks good

The last item was optional but “nice”. I found that wxPython fit the bill remarkably well, as it is based on GTK. If your list of constraints closely matches the above, you should give wxPython a look.

wxPython Qualities

At the time of this writing, wxPython is in release 2.1.13. Its author, Robin Dunn, has done a very good job of wrapping up wxWindows, one of the classic C++ GUI tool kits that has been around for several years. As a result of its relatively long existence and popularity, wxWindows has been ported to a number of toolkits and platforms and enjoys a relatively large widget collection. For Linux, the version of wxWindows that wxPython uses is wxGTK, the wxWindows interface to the GTK. For windows, the wxWindows port to win32 was used.

The interface of wxWindows to Python is remarkably transparent. The concepts are the same and the naming conventions are mostly the same. The object-oriented nature of the toolkit has been preserved. If you know how to program in C++ with wxWindows, you will have no problem with wxPython. Conversely, if you have a working proof of concept in wxPython, porting all or part of it to C++ for better speed should be straightforward.

wxPython is quite compact, but doesn't come with the standard Python distribution. See the Resources section on where to get the software. I found wxPython easy to learn; the distribution comes with a huge series of helpful demos which I will use in examples.

All of wxPython is open source, so people can contribute and fix bugs. In general, the software quality is high.

wxPython Drawbacks

For all its qualities, wxPython does have a few minor drawbacks. As of this writing, the wrapping of wxWindows is not complete, but Robin Dunn and others are working on finishing up.

The documentation is both very complete and almost non-existent. Robin can get away with this because wxPython is so similar to wxWindows that the wxWindows documentation, which is quite good and complete, can be used instead. The Python-specific part of the documentation is reduced to a small number of notes within the text (if a feature is not available or slightly different, for example) and a small section toward the end of the documentation. It is hardly any effort to do the mental translation from C++-type calls to Python, but if you've never seen C++ code in your life, this can be seriously off-putting.

The product is still very much in development. It follows wxGTK quite closely, but the wxGTK version it is based on is an unstable, development version. This means a number of things may not work perfectly, and things are likely to change a lot from version to version. I've had a lot of problems with printing under Linux, whereas it works perfectly under Windows.

wxPython would probably benefit greatly from a step-by-step tutorial. By the time you read this, someone will most likely have written one.

An interactive GUI-building tool would also be of interest. I understand that reusing the one from wxWindows should become possible soon.

Installation

The installation of wxPython is fairly straightforward if you have a Debian or an RPM-based system, as Robin Dunn provides both of these packages on his web page at http://wxpython.org/. You'll need wxGTK installed (again, a package is provided) as well as Mesa-3.0.

Examples

We are now going to look at some simple wxPython examples. The intention is not to offer a tutorial but to get a feel for the way wxPython can be used.

Getting Started

The obligatory “hello world” application is shown in Listing 1. A fixed-size frame is created at line 14. A panel is attached to the frame at line 18, and within the panel, a static text area displaying “Hello world of wxPython” in the default font is created in lines 19-20. The application subclass definition is shown on lines 23-32. An instance of our frame is created, shown and brought to the

foreground. If these definitions are run stand-alone, rather than being imported in another application, an instance of an application class is defined and run in lines 36-37.

Listing 1.

Note the systematic use of the object-oriented framework even in this very simple example. The frame and the application are subclasses of the pre-defined classes. We need to redefine only the methods we use to get our program working. A screen shot of this impressive application is shown in Figure 1.

Figure 1. ScreenShot of the "Hello" Application

One might say that about 40 lines of code is too many for a simple application that basically does nothing. In fact, you could probably cut this down to a few lines, but the goal is to present a relatively well-structured piece of code which is easy to build upon.

There are many function calls with **-1** given as second argument, such as **panel = wxPanel(self, -1)**. In these cases, the value of **-1** is a mandatory numerical identifier for the object being created, which is used when connecting widgets together, as we will see in the second example. A value of **-1** means "default" and is useful when this object ID will not be used anywhere in the code.

Slightly More Complicated Example

In the previous example, the position of the widget in the window was fixed, and we had no control over the application other than through the window manager: no menu, no exit button, etc. Moreover, getting the "hello world" message is a bit boring, so here's an application that does more and is much longer. The application is shown in Listing 2. In this second example, the structure of the program is essentially the same: we have a subclassed **Frame** and a subclassed **Application**. The subclassed Frame redefines the **__init__** function. In the redefined __init__, we define a few more interesting messages than "hello" on line 13-18, a counting variable on line 19, a menu table on lines 20-25. As before, we call the parent's __init__ function on line 27, create a panel on line 30, some static text field on line 31 and a button on line 34. Lines 36-43 are devoted to the layout of the main part of the window. Basically, we have the

static text in a fixed position; underneath it is a button, at the bottom of the window, that remains centered with respect to the right and left part of the window, no matter how wide the window is. This is done using a number of **wxBoxSizer** objects. On line 45, we associate a button event between the button defined on line 42 and the **OnButtonClick** method defined on line 69. This method will be called when the button is pressed.

Listing 2.

On line 47, we associate the application close event with the **OnCloseWindow** method, defined on line 76. Using this method, we build a quick message dialog box to ask the user for confirmation. On lines 48-64, we create a status line where the menu tool tips will be displayed. The main menu of applications is defined by the specs in **myMenuTable**, lines 20-25. Note that the "New" menu item will also call the method of line 81. In the definition of the menu items (lines 24-29), the character & before a letter indicates a keyboard shortcut (typing **ALT** + the letter calls the menu item).

On lines 65-67, we make the window layout fit together and set the autofit feature of the wxBoxSizer objects "on". Lines 69-72 define the callback method from both the menu item and the button. This method simply redefines the text content of the static text object defined on line 31. Pressing the button multiple times shows the strings defined in **self.myFortunes**, line 13-18, in succession. The method of lines 87-88 is called from the "Exit" menu item and invokes the **Close** event, which will close the application. Lines 90-106 are identical to the "hello" example. A screen shot of this application is shown in Figure 2.

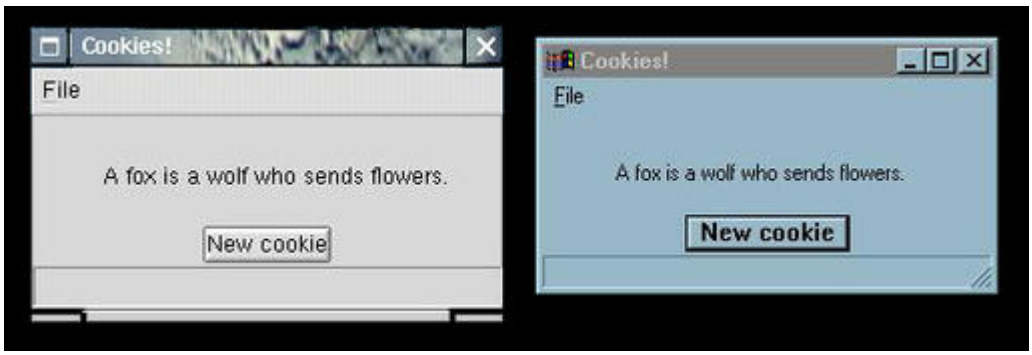


Figure 2. ScreenShots of the "Cookie" Application on Linux (a) and Windows (b)

The more interesting points of this "toy" application are how window layout can be achieved (using wxBoxSizer) and how events and widgets are connected (using **EVT_MENU** and **EVT_BUTTON** in this case). In fact, as it is, only the layout of the button is satisfactory; the main text is still in a fixed position—a complete solution would be too long to show here.

Note the use of the **self.*something*** variables. If a variable is to be accessed across methods, it must be called in this way (**self** being the object instance). If a variable is private to a method, the **self** prefix is not necessary.

Where to Go Next?

As mentioned above, a complete tutorial to wxPython is not ready yet, although one is on its way. Nearly all the widgets supported by wxPython are demonstrated in the demo application that comes with the distribution. Since the examples are concise and to the point, it is possible to learn from them fairly quickly.

Conclusion

At the moment, we are enjoying a growing array of usable tool kits, and I feel wxPython is one of the better ones. Like most things written in Python, programs in wxPython tend to be compact, easy-to-read and maintain and powerful. The range of available widgets doesn't match MFCs (Microsoft foundation classes), but is adequate for most applications. I personally find that this particular tool kit makes the RAD (rapid application development) concept finally available to Linux. The fact that applications written in this framework also work on Windows will be a tremendous benefit for some.

As it stands, wxPython probably cannot replace Tkinter as the default Python GUI tool, at least not until the Apple Macintosh port of wxWindows is completed and a stable release is made. But it shows a lot of promise, as being a much higher-level toolkit.

Resources

Credits

email: hugues.talbot@cmis.csiro.au

Hugues Talbot (hugues.talbot@cmis.csiro.au) is a Frenchman living in Australia. As such, he likes to eat, drink, cook and annoy his colleagues. His wife is constantly reminding him to get a hobby that has nothing to do with computing.

LINUX JOURNAL

Economical Fault-Tolerant Networks

Ali Raza Butt

Jahangir Hasan

Kamran Khalid

Farhan-ud-din Mirza

Issue #74, June 2000

We present a software solution which achieves fault tolerance by capitalizing on redundant replication of data and elimination of any single point of failure and with transparent switchover.

In the past decade or so, we have seen the evolution of computer networks as a merger of the communication and computing worlds. This revolution has made the dream of global connectivity possible, and with it has brought an explosion in the number of people utilizing computer networks for numerous vital tasks. The sheer volume and importance of tasks being ported to network-based applications and services has made reliability an undeniable need. On the other hand, networks fail all the time for a variety of reasons, including but not limited to power failures, communication link breakdowns, hardware failures and software crashes. Overcoming these failures and maintaining network services to users is addressed by implementing fault tolerance.

One common method for implementing fault tolerance is redundant data replication. Employment of an exact copy in place of a failed network component ensures availability of services on the network. The process of fault detection and replacement must be quick and automatic in order to make the fault invisible at the client end.

Our solution bases itself on a cluster of identical Linux servers, providing various network services. We did not employ any additional links other than the standard network connectivity. Various components within the solution include an algorithm to select a successor to a failed component, and both clock and data synchronization procedures among the processes. The services and data

are replicated on all computers within the cluster. In the event of failure of a server, it is to be replaced by a replica, which carries a redundant copy of all data and service configurations offered by the crashed machine. After the detection of a failure, an election process is enacted among the identical servers and a replacement for the crashed server comes forward. Under normal working conditions of the network, the current server regularly pushes clock time, data and key configuration files to the rest of the cluster in order to maintain them as peers.

Terminology

The Problem

The standard techniques for improving the reliability of the network can be divided into two broad groups.

- Dedicated hardware: this is expensive, and needs special maintenance. Examples of such a method include RAID (redundant array of inexpensive disks) and non-IP solutions requiring secondary communication links between machines.
- Secondary backup machines: these are machines identical to the primary server and are maintained as mirrors of the primary. Their major drawback is not their high cost, but that under conditions of network failure, the switchover to the backup machine is not transparent to the clients. In fact, the clients need to be configured in advance with secondary (or slave) server parameters. The clients for most of the services will usually try the primary server first; then, on timeout, attempt to use the secondary server. This introduces unnecessary overhead and delay. Another important thing to note is that the backup machine is 100 percent redundant, used only when the primary server is under fault.

The problem is in achieving a software solution to ensure reliability, without the need of additional hardware and to keep the switchover transparent to the clients. Furthermore, we would not like to waste resources by completely dedicating one or more machines to a backup role.

Proposed Solution

Our solution finds its roots in the already-established backup system of primary and secondary servers. Instead of using two computers, we employ a set of computers. One of these computers is selected to act as a master machine to coordinate the network services, while the rest assume the role of slave machines: general-purpose server-class machines that take part in the election. The master server sets up a virtual IP address and starts all services required for normal operation of the network. The slaves monitor the status of the

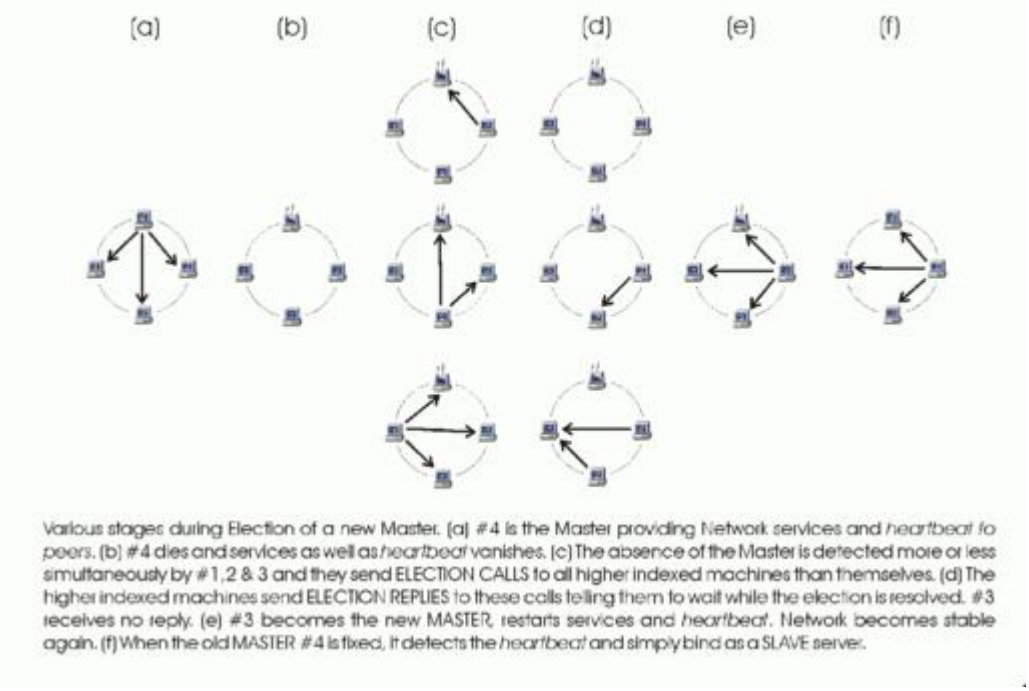
master for failures. If a failure occurs, a new master will be chosen and services to the network restored; the change is transparent to the clients. The slaves are not intended to be dedicated solely to this purpose; rather, they may be employed to perform other tasks (compute servers, workstations, etc.). Only on being chosen master does a machine also take on the task of establishing the virtual server.

Selection of the Master

The most crucial task in the event of failure is the selection of a new master machine. It is not possible to predict which machines in the chain would be unavailable due to earlier failures or being switched off. Therefore, a simple pre-established hierarchy of switchover, like the one in primary/secondary servers, is not practical. Moreover, a single coordinator for selecting a master cannot be used, because failure of this coordinator would result in total failure of the network.

With these facts in mind, we used a distributed election algorithm to select the appropriate successor to the dead master. Election algorithms are widely used to select a coordinator process in parallel and distributed computing. Initially, we tried the “Bully Election” algorithm. (See Resources.) This suffers from a handicap in that when a failed master is fixed and brought back into the network, it bullies the already-established master into handing over services. This creates an unnecessary switchover and results in loss of newer and updated data, besides causing network delay. The data loss occurs because updates may have been made in the new master machine, and when the original crashed master revives, it does not have the updated data.

Modified Bully Election Algorithm



Various stages during Election of a new Master. (a) #4 is the Master providing Network services and *heartbeat* to peers. (b) #4 dies and services as well as *heartbeat* vanishes. (c) The absence of the Master is detected more or less simultaneously by #1,2 & 3 and they send ELECTION CALLS to all higher indexed machines than themselves. (d) The higher indexed machines send ELECTION REPLIES to these calls telling them to wait while the election is resolved. #3 receives no reply. (e) #3 becomes the new MASTER, restarts services and *heartbeat*. Network becomes stable again. (f) When the old MASTER #4 is fixed, it detects the *heartbeat* and simply bind as a SLAVE server.

Figure 1. Election Algorithm

The “Modified Bully Election” algorithm is our modification of the basic Bully Election algorithm. Assume a stable network in which an array of servers is available. One of these machines is acting as the master machine and providing services to the network. The master server also multicasts a “heartbeat” signal to all slaves, apprising them of its existence. All the machines are assigned a performance index number. This number depends on various parameters such as processor speed, availability of latest data, available resources, etc. It may be dynamically assigned at each election, or in the simplest case, is an IP address. The higher the sequence number of a machine, the more it is suitable to act as the master server.

Now, assume a fault occurs, causing the master to crash. The rest of the machines detect its failure from the absence of the heartbeat and move into the election state. In this state, a machine sends election calls to all those with a higher index number than it has. If a higher-indexed machine is available, it will reply to this election call. On receiving an election reply, the machine moves into a “bind-wait” state. If a machine in election state receives no reply within a specified amount of time, it concludes that it has the highest index number. It then moves into “init” state. In this state, it starts the interrupted network services and resumes the heartbeat signal. It also sends a bind signal to other machines, moving them into slave state. Finally, it moves itself into master state, and the network becomes stable again.

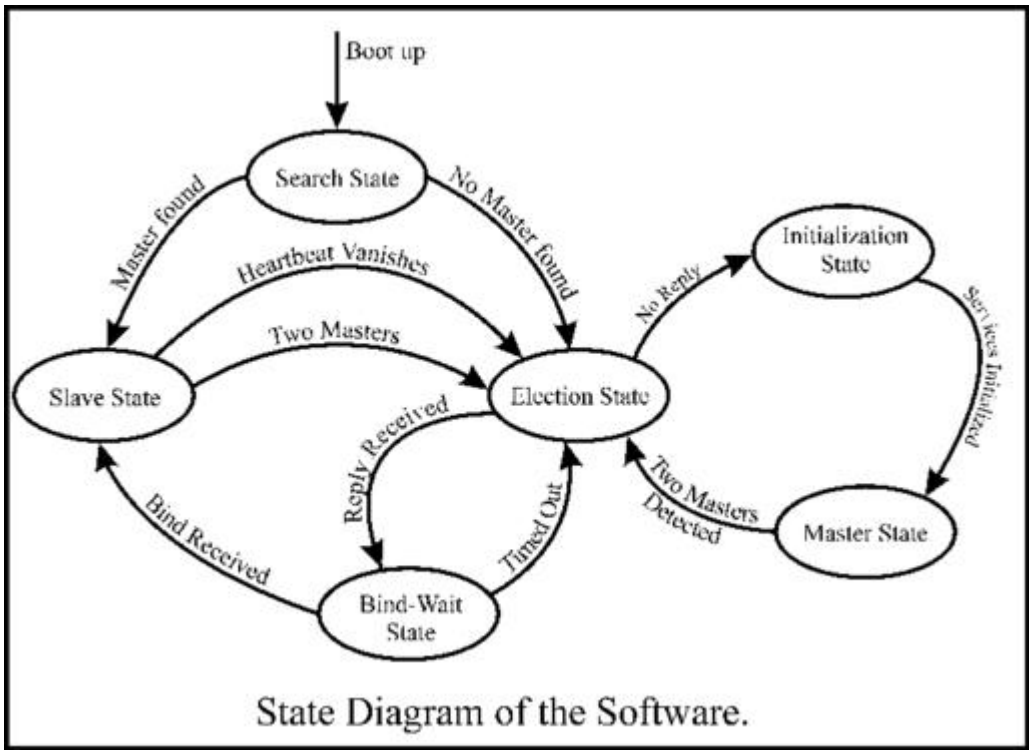


Figure 2. State Diagram

After receiving election replies, if the lower-numbered machines are unable to reach the slave state within a specified time, the election is timed out and restarted. Hence, this distributed process will elect a master under any sequence of failure. If the failed master is now fixed and brought up again, it will not initiate an election. In fact, it starts by listening to the heartbeat. If a master is found, it simply moves to slave state until the next failure occurs.

Our modification of the original Bully Election algorithm is that the computer with the highest index number is not necessarily the master at all times. In the original algorithm, a higher-index-number machine will always call for re-election and take control whenever it appears, regardless of the state of the network. In our case, this bullying procedure was found to be impractical for actual implementation. We implemented a variation, such that a stable network with a master will not be disturbed when a higher-indexed computer revives. It detects the presence of a master during startup, and on finding one, moves into slave state. Only if a new election is initiated will the highest-indexed machine take over. The current master is always the winner of the most recent election, and thus of the highest-indexed machine alive at that time but not necessarily at the current time.

The Two-Master Problem

A direct consequence of not including the bullying part in the election algorithm is the possibility of the existence of two masters. As we are relying on the master to take control of a key IP address as virtual server, two masters are

simply unacceptable. Such a problem may occur when a master is isolated due to network congestion, the other machines conclude it dead and elect another master. As the congestion resolves and the original master reappears, there will be two masters. A very simple technique to overcome this is that all machines monitor the source of the heartbeat signal. If it is not found to be unique, a re-election is done, resulting in a single master once again.

By following this algorithm, we are able to facilitate the infallible presence of a working master server on the network. Since it is a distributed implementation, failure of any single component does not affect the election. Hence, various services can be continuously provided to the network.

Virtual Server

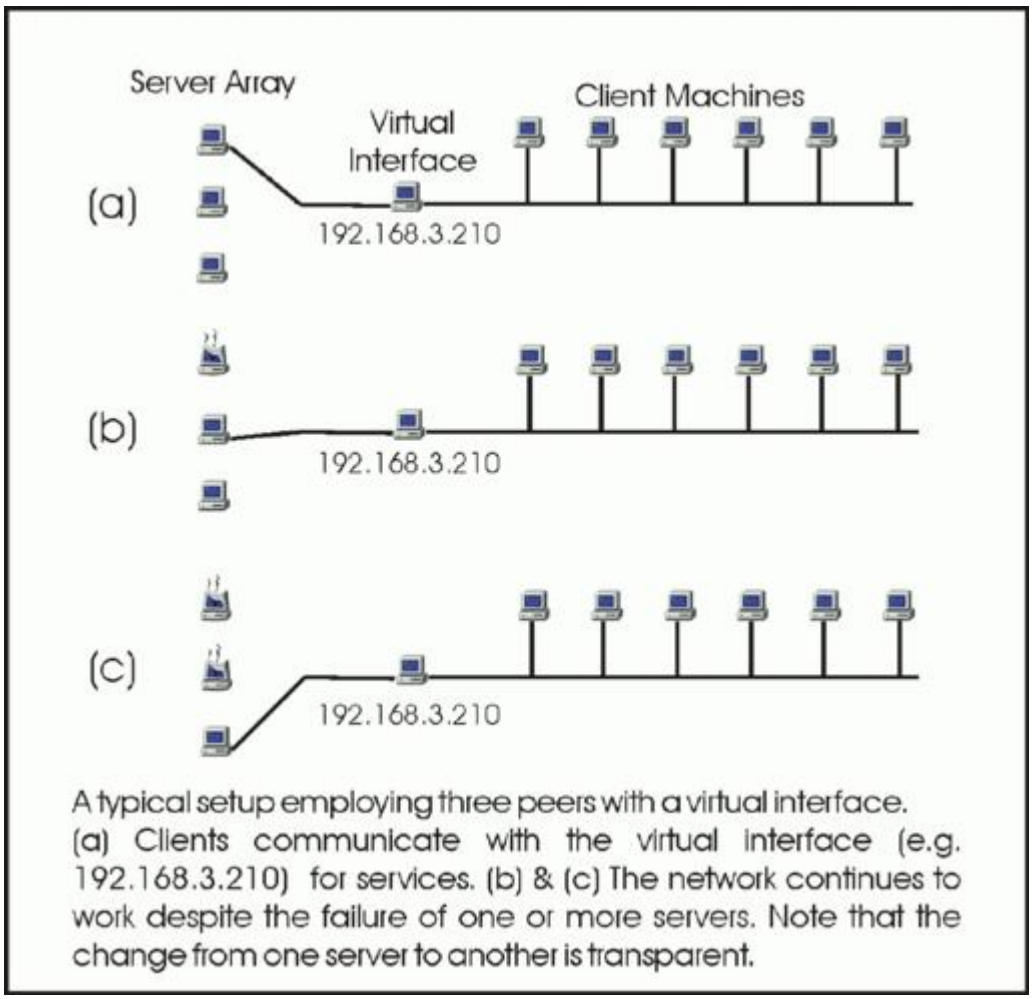


Figure 3. Virtual Server Diagram

The election process elects a master, but we are still far from making this change of masters transparent at the client end. In order to achieve transparency, we choose an arbitrary IP address henceforth referred to as the virtual server address. This address is not that of a machine in the cluster, but a separate one. All the clients are configured to use the virtual server address for

requesting services. Whenever a machine becomes master, it takes over the virtual server address and continues with its original. IP aliasing is used to achieve this, i.e., assigning more than one IP address to a single network adapter. The new master then starts providing the network services previously provided by the failed master. Thus, the virtual server is always available, powered by any of the machines taking part in the election process. Since the clients always communicate with the virtual server, the proceedings of the fault-tolerance algorithm remain invisible to them.

Machine Address Problem

The newly elected master server quietly takes over the virtual server address. However, the clients already have an address resolution protocol (ARP) cache entry connecting the virtual server IP address to the machine address (MAC) of the failed master. This cache would inhibit a client from communicating with the newer master, because the client would still try to communicate with the old MAC address. One solution which overcomes this problem is having an arbitrary MAC address be selected and taken over by elected masters. The problem with this approach is that not all network adapters support this function. Another solution would be to bluntly dump the ARP cache of all the clients and then reset the cache, which is also not an efficient technique.

The method we devised is to delete the virtual server IP address entry in the ARP cache of the newly elected master. Now the master automatically tries to update its ARP cache. In this process, it contacts the machines on the network, clients as well as slave servers. This not only updates the master's ARP cache, but also that of the clients. The advantage of this technique is that our software does not have to send special update packets to each computer—the already-working ARP mechanism does that for us.

Updating the ARP cache, followed by the IP address takeover, transparently causes the client to request services from the newly elected master. The clients may experience some delay while the actual election takes place, but other than that, they continue uninterrupted.

Maintaining Peers

A very critical perspective of the whole switchover scenario is that the machines should be maintained identically. Only then will the switchover become truly transparent. Steps must be taken to ensure that in the event of a failure, the likely new masters would have as much updated data as possible. Two important aspects of maintaining the peers are time and file synchronization.

Time Synchronization

Important and critical files need to be circulated on all the servers. Any server could have been a master, and might have newer versions of files. Thus, it becomes imperative that the servers be time synchronized, so that their file timestamps are comparable. This ensures that only the updated versions are distributed at the time of file synchronization. An important thing to note is that the time does not have to be matched with the real time. The only requirement is that all the servers have the same time. We relied on simply setting the clock to the time of the master server, using a remote shell procedure. No special time servers were used, although running NTPD or TIMED would have been a better technique.

File Synchronization

Another important task in maintaining peers is performing synchronization and replication of data over the entire array of servers in order to keep them consistently identical. The replication process is time consuming and often congests the network. The frequency of replication should be high enough to accommodate replacement transparency and minimize data losses during switchovers, while simultaneously low enough to allow proper network operation without undue congestion.

In a very dynamic scenario, it may not be possible to continuously distribute the updates on all machines taking part in an election. In such a situation, a switchover may cause a retrograde to the last synchronized version of files. Typically, the synchronization is scheduled during low workload hours. Additionally, instead of making backups, data is now distributed to the server array which better serves the purpose.

Implementation

Having discussed various necessary aspects of the software solution, we move on to its description. We implemented this solution using Perl 5.0 running on Red Hat Linux 6.0. Owing to the portability of Perl, the software runs on any version of Linux/UNIX with minor or no changes. The program is implemented as a daemon that is initiated at startup of the servers. It moves to the background after spawning four processes:

- Heartbeat listener process for processing heartbeat signals generated by the master server.
- Listener process for receiving and parsing various signals generated from other servers.
- Doctor process to interpret the heartbeat signal and decide whether the master server has failed.

- Elector process to actually implement the election algorithm and decide which actions need to be taken. It also generates a heartbeat signal if running on a master server.

The main daemon is supplemented with special scripts to handle startup and synchronization. Separate scripts are created for master and slave servers. This makes the configuration of startup services on both master and slave servers very easy.

Besides subordinate scripts, a host of scripts is provided for file synchronization and distribution. They may be initiated for scheduled synchronization and backup.

We employed UDP communication for heartbeat signals in order to minimize network load. For election calls and other signals, TCP is used to ensure reliability.

Security

If unchecked, any alien machine can generate an election call and the servers will move into an unwanted election state. For this reason, security and integrity of the signals exchanged between servers is highly stressed. We have implemented three levels of security for safe operation.

- Level 1: a list of all servers taking part in the election is maintained at all servers. Only signals from these machines are accepted; all other messages related to the election process are discarded.
- Level 2: the servers are state-oriented. That is, they acquire certain states, e.g., election state, master state, etc. In every state, some signals are anticipated and only these are accepted. Any other signal received, even originating from listed servers, is discarded.
- Level 3: an encryption scheme is used to encrypt all the signals. A random key is created for encryption and is valid for only one message. Therefore, even if a signal is intercepted and cracked, the encryption key will not be valid at any other time.

Results

For our implementation, we empirically found that weekly synchronization of large data suffices, while the password databases are replicated on every election. Other scenarios may require a more frequent synchronization.

The level of reliability and fault tolerance of the cluster increases in proportion to the size of the cluster. Increasing the number of servers increases the maintenance time for synchronization and unduly lengthens the election

process. In our situation, we determined from experimenting that three to four servers are enough to guarantee a practical working solution.

On average, the slave servers are negligibly loaded, while the master servers are not loaded for more than 0.1% of the CPU usage. Network load is also very low, except during heavy synchronization, which is therefore run as a scheduled process.

Our test bed for this implementation is the Digital Computer Laboratory, UET, Lahore, Pakistan. The lab consists of 10 Pentium-based servers and 60 diskless workstations connected by 10MBps Ethernet.

Future Directions

There is a need for development and implementation of techniques that provide for an immediate synchronization. One method could be that whenever a file is updated, all the servers update their versions of the file. In this way, all data on all servers at all times is perfectly synchronized, thus eliminating heavy network/server loads during scheduled synchronization.

Conclusions

This technique is a practical and feasible implementation of fault tolerance for low-budget LANs running open-source operating systems, such as in developing countries and resource-scarce academic institutions where expensive commercial solutions are just that—expensive.

Acknowledgements

Resources

Ali Raza Butt (drewrobb@mediaone.net), is a final-year student of EE-Communication Systems at UET, Lahore, Pakistan. Ali and the other authors are project group fellows for implementing this work and system administrators of the computer lab. Ali loves to program for network management and build PC-controlled gizmos.

Jahangir Hasan is a final-year student of EE-Communication Systems at UET, Lahore, Pakistan.

Kamran Khalid is a final-year student of EE-Communication Systems at UET, Lahore, Pakistan.

Farhan-ud-din Mirza is a final-year student of EE-Communication Systems at UET, Lahore, Pakistan.

Archive Index Issue Table of Contents

Advanced search

PoPToP, a Secure and Free VPN Solution

Matthew Ramsay

Issue #74, June 2000

When the expense of a remote access server is no longer attractive, it's time to look at the solution offered by a VPN.

Traditionally, remote access for employees has been through dedicated lines or a remote access server (RAS). A RAS typically consists of a collection of modems and telephone lines connected to a central machine. RAS can be quite reliable and secure, but it is expensive in its setup and long-distance-call costs. A Virtual Private Network (VPN) offers a secure, flexible and cheap solution in place of RAS and dedicated lines. PoPToP, the PPTP (point-to-point tunneling protocol) VPN solution for Linux, is a free VPN solution that businesses can take advantage of now.

VPN

A virtual private network is a private network capable of communicating over the public Internet infrastructure with a defined level of security. VPNs can exist between two or more private networks, often referred to as a server-server VPN, or between individual client machines and private networks, often referred to as a client-server VPN (see Figure 1). VPNs overcome the need for expensive dedicated lines or RAS dial in call and setup costs.

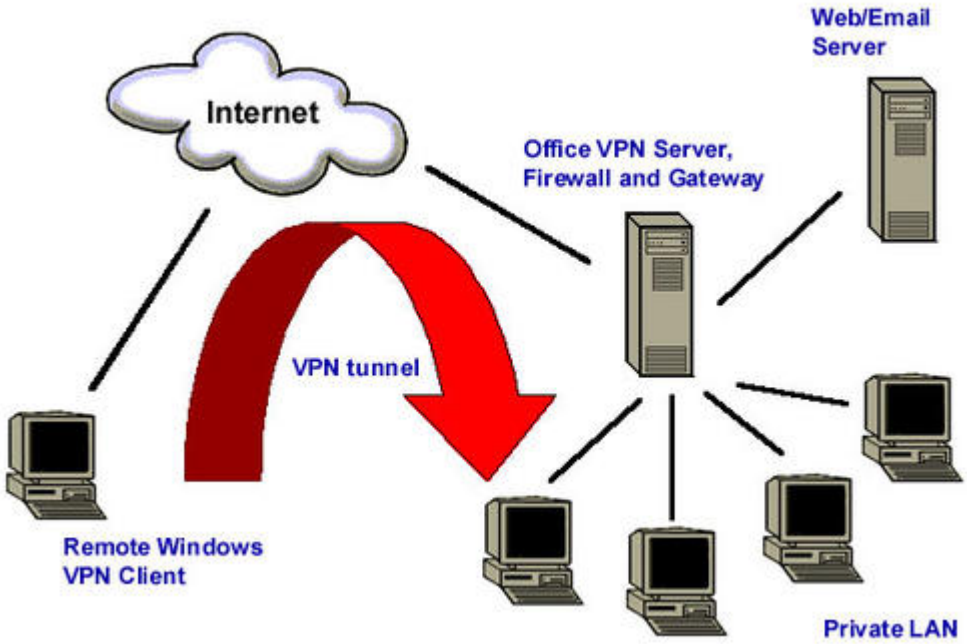


Figure 1. Example Client-Server VPN

In Figure 1, the remote client is handed a real IP address from their local ISP. This remote client can log into the VPN server, and hence gain access to the private network behind the firewall. The remote client can then browse and use other network services on the private network as if it were a machine on that network.

VPNs may also exist between multiple private networks (server-server VPN). For example, suppose your company has an R&D office in Australia and a sales and marketing office in the United States. Both locations have private networks that are connected to the Internet (the method, modem, DSL or something else, is transparent to the VPN). Traditionally, if the offices wish to share files on their networks, they would either have to e-mail the files to each other, dial in to each other or have some form of dedicated link between them. VPNs offer a cost-effective solution for joining these two networks seamlessly, without compromising system security.

Different Types of VPNs

The most popular VPN technologies available today are PPTP and IPsec. Much debate and analysis has occurred recently between proponents of these competing VPN technologies. Both PPTP and IPsec have an important role to play in VPN solutions. But neither PPTP nor IPsec is without flaws.

PPTP is an open-documented standard published by the Internet Engineering Task Force (IETF) as RFC 2637, available at ftp.ietf.org/rfc/rfc2637.txt.

The operation of PPTP as a VPN is performed by encapsulating the point-to-point protocol (PPP) in IP and tunneling it through an IP network. All communication, authentication and encryption is handled almost exclusively by PPP, which currently supports PAP, CHAP, MSCHAP and MSCHAPv2 authentication. PPP encryption is performed through compressor modules, and available patches under Linux allow PPP to support RC4-compatible 40-128-bit encryption. Some people make the mistake of assuming that since PPTP uses PPP, you need a modem. This is not the case. In fact, the connection mechanism to the IP network is transparent to PPTP.

PPTP is widely deployed in both client and server forms due to its default existence in Microsoft Windows platforms.

IPsec

IPsec is a new series of authentication and encryption security protocols that can be employed for sending data securely over IP networks. IPsec offers encryption, authentication, integrity and replay protection to network traffic. IPsec also specifies a key management protocol for establishing encryption keys. IPsec, like PPTP, is an open standard developed by the IETF.

PPTP vs. IPsec

PPTP is transparent to the authentication and encryption mechanism. Microsoft's version of PPTP was recently upgraded to include MSCHAPv2 and MPPE-enhanced (and more secure) security protocols. Patches are available for the Linux PPP daemon that allow PPTP solutions such as PoPToP to take advantage of Microsoft's enhanced VPN security.

Bruce Schneier, Chief Technical Officer of Counterpane Internet Security, Inc., and perhaps the chief guru of Internet security, recently analyzed Microsoft's MSCHAPv2 and MPPE security protocols. Schneier concluded that this release of MSCHAPv2 from Microsoft addressed the major security weaknesses found in MSCHAP.

IPsec was also recently analyzed by Schneier (with the help of Niels Ferguson). In their analysis, they concluded that IPsec's complexity effectively makes it impossible to implement a secure solution. They believe IPsec will never result in a secure operational system. They emphasize that although IPsec has its flaws, it is a more secure solution than PPTP.

IPsec remains a new technology, and future improvements are sure to enhance its security further and increase its attractiveness to business. Additionally, with its default presence in Windows 2000, IPsec will offer small to medium-sized businesses a more secure and affordable solution.

Affordable PPTP VPN (with MSCHAPv2 and 40-128-bit RC4 encryption) is available now. With the countless Windows machines already out there supporting PPTP VPN, the cost-effective solution is obvious. Windows 98 has VPN client software as an install option. Windows NT 4.0 comes with PPTP (server and client) by default. Patches (Microsoft Dial-up Networking patch) exist for upgrading Windows 95 machines to include a PPTP client. Windows 2000 has PPTP by default.

The Free PoPToP

PoPToP is the PPTP VPN server for Linux. Ports exist for Solaris, OpenBSD, FreeBSD and others. PoPToP allows Linux servers to function seamlessly in PPTP VPN environments, enabling administrators to leverage the considerable benefits of both Microsoft and Linux. The current release version of PoPToP supports Windows 95, 98, NT and Windows 2000 PPTP clients, as well as the Linux PPTP client.

PoPToP is a PPTP access concentrator (PAC) that employs an enhanced GRE (generic routing encapsulation—protocol 47) mechanism for carrying PPP packets, and a control channel (port 1723) for PPTP control messages. The basic operation of PoPToP is to wrap PPP packets up in IP and send them across the public Internet infrastructure. At the other end of the connection, the PPP packets are stripped from their IP packets and handed to the PPP daemon. The operation is almost identical to a dial-in session, except the PPP packets are wrapped in IP and sent over an IP network as opposed to a generic phone line and modem configuration.

PoPToP can be set up to work with a patched PPP daemon to support MSCHAPv2 authentication and RC4-compatible 40-128-bit encryption. A Linux server running PoPToP can effectively replace a Windows NT PPTP VPN server. However, PoPToP does not support PNS operation, so it does not replace a Windows NT server when PNS is required.

Another advantage of PoPToP (and PPTP in general) is that it is transparent to the encryption and authentication mechanism. Porting an alternate encryption algorithm (such as Blowfish) to a PPP compressor module would not be a difficult task. The only issue with developing your own encryption and authentication mechanism is the simple fact that you will break generic Windows client support. However, the Linux PPTP client is available under the GNU GPL and will work seamlessly with any PPP changes.

Finally, PoPToP is simple. It has a tiny memory footprint and has undergone performance tweaks. This makes PoPToP very attractive to embedded platforms and edge networks. When teamed up with the Linux PPTP client,

solution providers can offer cheap VPN solutions with their own defined security protocols.

PoPToP was originally pioneered by Moreton Bay (http://www.moretonbay.com/) in February 1999 for their eLIA (embedded Linux Internet appliance) platform. It was released under the GNU GPL in April 1999, and has since found widespread acceptance on standard Linux servers and firewalls in both large production sites and small business and home networks. PoPToP is in the current Debian "potato" code freeze and SuSE 6.2.

Setting Up PoPToP

Setting up PoPToP with a standard PPP daemon (without MSCHAPv2 or RC4-compatible encryption) is a painless task. Below is a quick setup guide.

- Grab the latest stable version of PoPToP from www.moretonbay.com/vpn/download_pptp.html.
- Log in as root to install and run PoPToP.
- If you downloaded the PoPToP v1.0.0 tar file and stored it in /usr/local/src/, type the following commands:

```
cd /usr/local/src/
tar zxvf pptpd-1.0.0.tgz
cd pptpd-1.0.0
./configure make
make install
```

- If you downloaded the PoPToP RPM (pptpd-1.0.0-1.i386.rpm), type the following:

```
rpm --install pptpd-1.0.0-1.i386.rpm
```

- PoPToP's binaries are placed in /usr/local/sbin. Check to make sure **pptpd** and **pptpctrl** are there before continuing.
- Set up PoPToP configuration files. Example configuration files are shown in Listings 1 and 2. This configuration will use CHAP as the authentication mechanism. The user login is "billy" and the password is "bob".
- Now that the configuration files are set up, you are ready to launch PoPToP. Simply type **pptpd**.

Listing 1

Listing 2

Any standard Windows client with PPTP VPN installed should now be able to connect to your PoPToP-enabled VPN Linux server. On Windows 98, you can

install it via Control Panel-->Add Remove Programs-->Windows Setup-->Communications-->Virtual Private Networking.

Conclusion

Remote access for employees no longer needs to be an expensive process. VPNs can easily replace dedicated lines or remote access servers without compromising security. PPTP is one VPN technology that is ready now. Although criticized in the past for its security flaws, recent enhancements to the authentication and encryption protocols have made PPTP an attractive solution to business. PoPToP is the PPTP VPN solution for Linux that can take advantage of MSCHAPv2 and RC4-compatible 40-128-bit encryption available to the countless Windows client machines. PoPToP is easily installed and is free. PoPToP is simple, and due to its small memory footprint, very attractive to embedded platforms and edge networks.

Resources

Glossary

email: matthewr@moreton.com.au

Matt Ramsay (matthewr@moreton.com.au) is a full-time, low-level Linux application hacker living in sunny Brisbane, Australia. In addition to PoPToP, he has also developed and released under the GNU GPL a “micro” DHCP server for Linux. His main professional focus is writing small and fast applications for embedded Linux systems.

Archive Index Issue Table of Contents

Advanced search

Linux for the End User—Phase 1

Clay Shirky

Issue #74, June 2000

Is Linux ready for the casual user? Mr. Shirky begins an experiment to find out.

Several months ago, I bought some clothes with my LinuxFund credit card. The guy behind the counter looked at the card and said, “Linux! That's the future, right?” He then leaned over the counter and asked “What do you think is going to happen to the PC?”

The biggest single change in that loose amalgam that is the “Linux community” in the last year is that it now includes people, many thousands of people, who have heard about Linux in the press and liked what they've heard, but don't know anyone who uses it. The old model for the spread of Linux—you tell two friends and they tell two friends—is over, because the press attention has spread news of Linux *far* outside the community of actual users, to the point where a random cashier in an unrelated business feels confident of its revolutionary impact, even without any direct experience with the OS itself. For these interested-but-unconnected people to become Linux users, they will have to work harder than any of the Old Guard did, because they don't know anyone with enough experience to help them. They will have to find their way to Linux more or less from scratch.

I was reminded of my encounter with the Linux-loving cashier a few weeks ago when Almaz, my wife, decided that she was writing too much to keep working on her laptop and wanted a desktop. All she needed to do on the desktop, she said, was use Netscape, WordPerfect and a printer. And so an experiment was born, an experiment to see whether, armed with only a credit card and a willing Windows user, one could buy a complete Linux desktop system with a printer which that user could set up and get started with on their own. Almaz, who uses a computer every day but has no burning desire to understand what goes on under the hood, seemed to be the ideal test candidate.

This experiment is really two questions rolled into one. The first obvious question is, “Have the recent improvements in usability created a Linux which a casual user would feel comfortable with?” The second and less obvious question is, “Are there companies thinking about how to package and deliver such systems to interested home users?” Since I needed to procure the system before testing it, I set out to answer the second question first.

Phase 1—Needle in a Haystack: Finding a Linux Vendor

To begin, where to find this mythical, pre-configured desktop system? Although I have bought several systems with Linux pre-installed from several different vendors, they were all web servers. In this case, looking for a desktop system, I was almost as much of a newbie as my wife was. I set out down three different paths—using a search engine (Yahoo!, in this case), going to the major Windows PC vendors to see if any of them would ship a PC with Linux, and going straight for Linux URLs.

Search Engine

Reasoning that average users would have little idea of where to turn for pre-installed systems, I went straight for Yahoo!, usually a beacon of clarity in the fog of the Web. Yahoo! did not disappoint in this regard, with a manageable 22 categories which included the word “Linux”. But of these, only three categories were links to hardware vendors. Furthermore, two of those three were specifically business-to-business, leaving me only the slightly off-putting “/Hardware/Custom_Built_PCs/Linux/” as a category.

The problem with this category is that there's no such thing as a “custom-built” computer anymore—in the age of Dell, *all* computers are custom-built. Customers of commodity PCs expect to be able to adjust the configuration to their taste, and they don't think of this as buying a “custom-built PC”; they just think of this as buying a PC, period. A big part of serving interested but inexperienced users is creating a sense that they are not going to be bombarded with too many choices, but no one selling Linux systems has figured that out yet.

Yahoo! included a scant nine sites which advertised Linux hardware, and these were not always listed in the most user-friendly terms. Would you send a want-to-be Linux user to a site advertising itself as “a Linux-centric systems integrator offering support, hardware, applications, development and services”? Nevertheless, these sites went on the master list. Time for the next strategy: going straight for the traditional PC retailers.

The Majors

The second strategy I pursued was going after existing PC vendors. I had heard that Dell, Compaq, HP and IBM had all “embraced” Linux, though I wasn't really sure what that meant and visiting their web sites provided few additional clues. While a search for “Linux” on all four sites turned up thousands of pages of documents, FAQs and press releases talking up their commitment to Linux, I was unable to find out, using only their web sites, whether they would sell me a Linux desktop system or not. In all four cases I had to resort to the phone, and in three of the four cases, Compaq, HP and IBM, I was told they sell Linux only on servers.

Dell, as usual, was a bit further along than its competition, and sells high-end Linux PCs, but the base price is over $2000, too high for a casual user's desktop. Furthermore, this option isn't even listed in the Home and Home Office section, only in the Business Users section, which would be daunting for newbies.

Nevertheless, if desktop Linux becomes a reality, Dell will be the company to beat. In my conversations with them, they always put me in touch with knowledgeable staff and they never sugar-coated anything, telling me, for example, that printer setup was likely to be “quirky”. I opted not to buy what was really a business-class workstation for Almaz, but when and if Linux for the desktop becomes a reality, Dell is better positioned to take advantage of it than any other Windows PC vendor. Everything they do is geared toward making the customer feel comfortable about making a decision.

So it's a strike-out with the traditional PC vendors. Time for the last strategy: going straight for Linux URLs.

Linux URLs

In terms of newbie reach, what's the most valuable piece of Linux real estate on the Web? (Hint: it isn't Slashdot.) It's Linux.com, the Linux portal owned by VA Linux Systems, because end users typically think of dot-com sites as the Web's anchor tenants. Linux.com is a beautifully simple piece of web design, but as with so many sites, it suffers from insideritis. Clicking on “Desktops”, for example, gets you a page with a bewildering array of conversations about KDE vs. GNOME, but nothing at all about how you could buy a Linux desktop. Clicking on “Get Linux” gets you a page of every Linux distribution out there, but again, no information about actually getting a computer running one of these distributions. Searching for the word “PC” brings up page after page of BBS flames about Windows 98 vs. Linux, but yet again, nothing for the user who wants to buy a Linux PC.

Linux.com has no obvious, Yahoo!-like list of Linux system vendors. I have no way of knowing if this is a simple omission, or if VA Linux refuses to list competitors on a site it owns, but it seems like a huge oversight to host the single most valuable Linux URL from the outside world's point of view without directing interested users to places where they can actually buy a computer running Linux.

The problem here is that ordinary users don't buy operating systems. They buy computers. If there is ever to be a real spread of Linux to casual users, it will mean a move from a focus on this or that Linux distribution to a focus on pre-installed systems. Vendors will have to drop the idea that someday the average user will be comfortable installing an operating system—they don't even like to install ordinary software, so no matter how easy an OS install can be, it will never be as easy as buying it pre-installed. As the size of the Linux user base grows, the number of people who are comfortable installing from scratch will grow in absolute numbers, but will shrink dramatically in total market share. Whoever focuses on selling Linux computers rather than Linux distributions will win in the long haul.

Abandoning Linux.com, I went on to Linux.org (though I am convinced that most end users would not take this step on their own), and finally hit pay dirt.

Linux.org has the most complete list of vendors of Linux systems I have found anywhere, with almost 80 vendors listed in the U.S. alone. However, the list is hard to use, as it's in no particular order and has spotty information about what the vendors do. Furthermore, it looks like a prototypical web site circa 1996, with lots of white space between entries for “readability”, meaning that you can't see more than a handful of vendors at any one time, wrecking the ability to do any comparison shopping.

Nevertheless, this (plus a few sites from the Yahoo! list) is clearly the best source of vendors I'm going to find. Phase 1 is now done: I have gone from a standing stop to a long list of Linux system vendors to check out. Elapsed time: 48 hours. Frustration quotient: medium to high. Amount of work required: much too high. Phase 2 will be evaluating the sites themselves, and Phase 3 will be road-testing the ease of setup and use for whatever system we buy.

As a side effect of looking though these sites, it became clear to me that as the user base spreads, pre-installation will trump ease of use for determining which flavor of Linux users choose, and by this measure, Red Hat is *way* out in front. No matter how easy the Caldera or Corel distributions are (the two easiest installs in my experience), the real trump card in terms of growth in user base will be deals with hardware vendors. Surprisingly, no home page for any Linux distribution I looked at (Red Hat, Corel, Caldera, Debian, Mandrake)

had what I thought would be an obvious link: a big red button that said “Click here to buy a computer running FooBar Linux from our partners”. The owners of different distributions are still assuming that most users will install their own Linux and, though that may be true for the moment, it will not be for long.

Conclusion: On a scale from “Rough” to “Smooth”, Phase 1 surely counts as rough. For someone interested in using a Linux box as a casual user, but with no idea where to turn, there is no obvious answer about where to look first. This bootstrap problem, where you have to know where to look before you do the looking, is the Achilles heel of easy-to-use Linux right now. While it is unclear whether Linux will ever make any real inroads into the MS desktop monopoly, if the software were ready tomorrow, interested users still couldn't find it. As we have learned from Amazon, eBay and Yahoo!, this means that if this space ever does take off, the first vendor to associate themselves firmly in the Web's collective mind as the official vendor of Linux desktops could walk away with a lion's share of the traffic. Next month, we'll evaluate Linux systems vendors.

email: clay@shirky.com

Clay Shirky (clay@shirky.com) is currently the Professor of New Media at Hunter College, where he teaches in both the undergraduate and graduate programs. He has worked as a writer, programmer, and consultant, for Business 2.0, FEED, Silicon Alley Reporter, word.com, Urban Desires and net_worker magazine.

Archive Index Issue Table of Contents

Advanced search

The Artist's Guide to the Linux Desktop, Part 3

Michael Hammel

Issue #74, June 2000

In this episode, Mr. Hammel tells us about the Window Maker window manager, a less flashy but more mature product than Enlightenment.

Window Maker is a descendent of the GNUStep project. GNUStep is, in a sense, more like GNOME and KDE than Window Maker. GNUStep attempts to provide underlying features of the desktop—features which are not visual, but functional. Window Maker includes some of this functionality, but is more akin to Enlightenment, a window manager, than GNOME, a desktop environment. Designed for additional integration support of GNUStep applications, Window Maker takes its look from NeXTSTEP, the desktop interface from the Apple-devoured NeXT company, one of Steve Jobs' post (and pre)-Apple projects.

Window Maker includes support for running as the window manager in KDE, GNOME and Open Look (yet another environment, although you aren't likely to see it on Linux). The KDE support appears more complete than the GNOME support (which tends to be a bit of a moving target at times). The source distribution contains README files describing how to use the window manager in both KDE and GNOME. Window Maker has some of its own session management capabilities, but does not directly provide drag-and drop-between applications.

Again, we'll assume you already have Window Maker installed. If not, instructions can be found in my article “Installing Window Maker” in “Strictly On-Line”, http://www.linuxjournal.com/lj-issues/issue74/4050.html. Be sure you have version 0.61 or the latest release.

Touring the Desktop

The first time Window Maker starts, you'll see a display similar to Figure 1. This is the default screen display, which isn't that fancy, initially. Unlike

Enlightenment, Window Maker doesn't presume any graphical pizzazz on your part. It lets you define it right from the start.

Figure 1. The Default Window Maker Display

Note that on my desktop, I use the XPM-based default icons (see Figure 1), whereas on my laptop, I used the TIFF-based default icons. I never could get the TIFF-based version, which has much nicer icons, to work properly on my desktop even though the desktop and laptop systems are configured—as far as I can tell—exactly the same. This is a negative mark for Window Maker. Neither FVWM2 (my normal window manager) nor Enlightenment exhibits this sort of problem.

Figure 1 shows three important pieces of the Window Maker desktop: appicons, the clip and the dock. Window Maker uses icons in two distinct ways: as appicons and as mini-windows. This is another terminology dance we users have to play, as developers struggle with ways to define the complex to the uninitiated. Don't let it worry you much. The images along the right side of Figure 1 are appicons. They represent a running application whose windows can be open or iconicized. This differs from a mini-window, which is what you get if you iconicized a window. Click on the small box on the upper left of a window, and the window collapses into a mini-window. Visually, you can tell the difference between a mini-window and an appicon, as the mini-window has a tiny title bar in it, as shown in Figure 2.

Figure 2. Appicon and Mini-window

Appicons (see Figure 2) are created each time you start a new application from a command line. This is different behavior from most other window managers, and I initially found it annoying. Fortunately, you can change this behavior by using the Dock, which I'll discuss in the section on managing the desktop.

In the upper left corner of Figure 1 is the Clip. This is essentially the pager for Window Maker, although it lacks the visual cues that other pagers provide. It also has some special behaviors when used with appicons. Dragging an appicon onto the clip connects the icon to the clip for the current workspace (see Figure 3). After a moment, unless you keep the mouse over the clip icon, the icon you just connected to it will be hidden. This allows you to use drag-and-drop to place applications on a particular desktop, plus it hides all those nasty appicons. The clip is a tool for managing which applications are on which desktop.

Figure 3. The Clip with Appicons “Attracted”

This leads to another issue: Window Maker doesn't have desktops. It doesn't have pages, either. Instead, it has workspaces. More terminology, but in this case, it's not needed. A workspace is the same thing as a desktop. Unlike E (Enlightenment) or FVWM2, Window Maker doesn't have multiple pages for each desktop. It just has an unlimited number of desktops connected in a sort of linear—left to right—pattern.

Dragging windows between workspaces is configurable using the Preferences utility (discussed in more detail later). Drag to the left or right edge to get to the next workspace. You can warp (i.e., jump) a window to another workspace by

using the window's Title Bar menu “Move To...” option. When you get to the edge of the last window, you can't drag past the right edge unless you configure in the Preferences utility to wrap back to the first workspace or to create a new workspace—either is configurable. The same thing applies when trying to drag to the left edge of the first workspace.

Adding new workspaces can be done by clicking with the middle mouse button (normally mapped to both buttons on a two-button mouse) on the Clip. This brings up a menu, the first item of which is “New”. A right mouse click on the Clip brings up a menu for managing the current workspace, from renaming it to grabbing all appicons to managing how icons are displayed.

When switching workspaces with the Clip, the name of the workspace you've switched to is shown in the center of the display (over any windows, if necessary): it then fades from view. This is a cute trick that happens to be a bit useful as well. It's nice to know the name of the workspace you just entered.

Over on the top right side of Figure 1, you'll see the Dock. This is a special application that manages appicons. This is the place to add appicons to launch applications. It's similar to the GoodStuff bar in FVWM2, or less similarly, to the panels in KDE and GNOME.

Using the Dock is a little confusing at first. Let's walk through an example of adding an appicon to the Dock, so we can start xterms at will. The first thing to do is open an xterm manually, which you can do by right-clicking on the root window to get the root menu. The third item down should be “XTerm”. Click on it. A generic black text on a white background xterm should open, and its appicon placed in the lower left of the screen. Left-click on the appicon and drag it up under the appicons under the Dock. When you're close enough, a white box is displayed under the icon, and when you let go of the mouse button, the xterm's appicon is neatly positioned under the other icons.

Now you have an icon that will always be there (you can remove it manually later) when you start up your X session. Right now, its application—the xterm—is already running. Exit out of the xterm by typing “exit” at the command prompt or clicking on the “x” in the upper right corner of the xterm window. Notice the xterm's appicon now has three small dots in its lower left corner. That means you can start an xterm by double-clicking on the appicon. If you do this now, the xterm opens again and those dots disappear.

The problem here is that you can't open more than one xterm. You have to configure the appicon, while it's docked, to allow for more than one instance of the xterm to be started. Here is how you do that:

- If it's not open, open an xterm by double-clicking on the docked appicon.
- Right-click on the title bar and choose “Attributes” from the menu.
- In the “Inspecting...” dialog, select the “Application Specific” option.
- Click on the “No Application Icon” button, then click on “Save”.
- Click on the “Close” button (the X in the upper right) of the “Inspecting...” dialog to close that window.

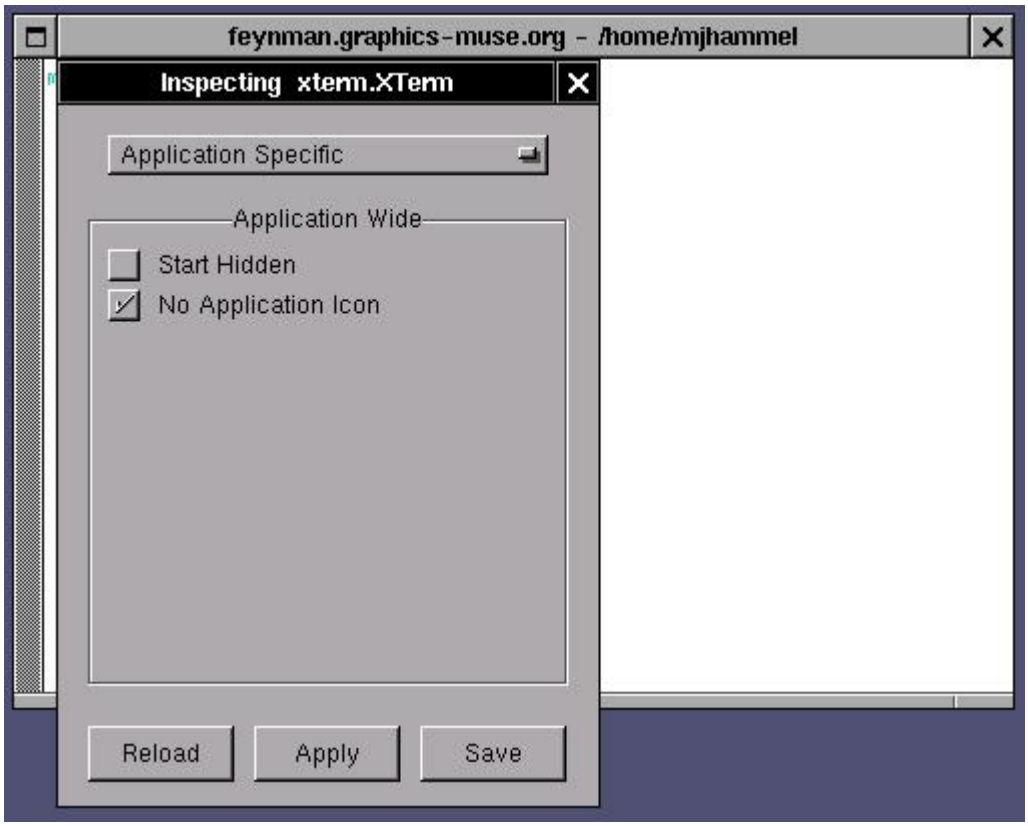


Figure 4. Turning Off the Application Icon

Now you should be able to launch multiple xterms from the single terminal icon hanging off the Dock. This process works for any appicon you connect to the Dock.

If you want to remove an appicon from the Dock, just click and drag it down to the bottom of the screen and let go; it will disappear. First, make sure you've closed all instances of the application started with that icon, however.

Figure 5. The Default Dock Applications

The default Dock applications are shown in Figure 5. The top one is the Dock itself. The next one down will launch generic xterms. The last one opens the

Window Maker Preferences utility. The icons you see may differ if you are using the TIFF-based icons instead of the XPM-based icons.

Root and window menus can be partially displayed off-screen depending on where you open them, but you can scroll them back into view by moving the mouse pointer to the edge of the screen (while over the menu) and holding it there for a moment. You can see this by right-clicking near the left or right edge of the display.

Figure 6. Example of Themed Root Menus

In the default root menu (see Figure 6), the entry “Preferences Utility” under “Appearance” didn't work for me. That's because I changed the default installation directory (using **--prefix** when I built from the source). To fix this, edit the file $HOME/GNUstep/Library/WindowMaker/menu and comment out the line for “Preferences Utility”. You don't actually need this if you use the Application Dock. The last icon on the dock in the default configuration, the one with the heartbeat line, will launch the Preferences Utility correctly.

Clicking in the title bar of a root menu will anchor it—the menu gets a close button in the upper right, and doesn't close when the other menus are closed. To close it, click on the Close button.

There is a tool in the Preferences Utility for managing the root menu graphically, but if you use it, you lose access to the text-based versions of the menus. The graphical tool converts the text-based menu into a sort of compiled format. For moderately intelligent users, the text-based menus are quicker to update, and for internationalized menus, you are required to use the text-based menus—internationalized menus can't be managed using the graphical utility. Making manual changes to the text menus does not require restarting Window Maker; the changes are recognized the next time you open the root menu. If you choose to use the graphical interface for menu management, its updates are also recognized immediately.

The default root menu is GNUstep/Library/WindowMaker/menu. This single file holds all the menus you see hanging off the Root Menu (the Applications, Editors and Miscellaneous menus, for example). To add or delete entries from these menus, just edit this file using your favorite text editor. The root menus will get updated automatically once you save the file back to disk.

Desktop Management

Now you know what you're seeing on-screen. The next step is to decide what to do with all of this. Many of Window Maker's features have equivalent counterparts in Enlightenment, such as shaded windows: double-click the title

bar to shade the window, again to open it back up. Windows can be grouped in E using manual configuration. In Window Maker, you have to select windows. Left-click in the root window, hold and drag over the windows you want to group. You can now drag these windows to another desktop. To unselect them, just left-click once in the root window.

Moving between desktops is easy: simply type **ALT-#**, where **#** is the desktop number as shown in the Clip. This is a fast way of bouncing around your desktops, something I can't currently do in my window manager, FVWM2. You can also move between desktops using the root menu Workspace submenu, the Clip menu, and, if it's enabled, by moving the mouse to the left and right edges of the display.

As you've seen, an important part of the Window Maker desktop is the Dock. It allows you to launch applications quickly. It can also be used by special Dock Applications, small programs (similar to E's epplets) that run right in the icon. Examples of these programs include clocks, system information, stock and weather reports and a chess game (see Figure 7).

Figure 7. Some Sample Dock Applications

Docked applications can launch multiple instances of an application using the technique described earlier. Some applications don't behave properly under Window Maker, however, and you can't disable their application icon, so that technique won't work. Never fear: there are alternate ways to launch multiple instances. The simplest is to use the Launch option in the docked appicons menus (right-click on a docked appicon). Selecting Launch will open another instance of that application. Alternatively, you can change the command to run in the “Settings” dialog, also from the docked appicons menu. For example, XV is normally launched as **xv**. This gets changed to

```
/bin/sh -c "exec xv&"
```

Now, a double click on the XV appicon in the dock will launch a new instance of XV every time.

There are many dock applications available for Window Maker. The primary web site for finding these is the DockApps Repository at http://www.bensinclair.com/dockapp/.

You can also find a number of interesting icons and themes plus various bits of useful documentation at http://wm.current.nu/.

Image is Everything

Window Maker doesn't permit quite as much variation to the window border system as Enlightenment does, but this actually simplifies the creation and use of themes. The last item in the default menu, “Appearance”, allows you to access some predefined themes and styles (essentially the same thing, from a user's perspective). The Preferences Utility is used to make specific changes to window borders and backgrounds, after which you select the Appearance-->Save Theme option to save those settings.

The Preferences Utility is not what you use to set the background, however. To set the background image, you use the **wmsetbg** program. Unfortunately, there doesn't appear to be a way to choose files randomly for setting the background built into the default configuration. You'll have to add entries to the root menus for each image you wish to use as a background. On the plus side, once you've set the background for a given workspace, you can save the setting so that it's always used in the future.

The Window Maker Preferences Utility (see Figure 8) is the heart of configuration for the look and feel of WM. All visual settings are configured here:

- Animations can be disabled here. Useful for memory-limited systems like laptops.
- Icons can be sized up or down from their default 64x64 size.
- You can configure extra paths to search for pixmaps and icons.
- Many keyboard shortcuts are configurable by using a graphical interface. This is very useful for someone like me. On my laptop for instance, I have one of those post mice (IBM-style red-capped post). I hate it, and I don't like moving back and forth to my real mouse on my desktop, either. I prefer just bouncing around using the keyboard when possible. Being able to configure these shortcuts easily is a welcome feature.
- The mouse is fully configurable, too. You can specify which mouse button will open menus, how fast the mouse responds to double clicks and so forth. This is a very useful but seldom provided feature for window managers. It's especially nice that it has a graphical interface.

Figure 8. The Window Maker Preferences Utility

The Window Maker Preferences Utility, known simply as WPrefs, is also used to configure the backgrounds, window borders, title bars and icons to use for specific windows. You can't get as creative as you can with E, but the interface for making changes is fairly sophisticated and rather intuitive. As I mentioned

before, the only thing it doesn't do is set the background image of the root window, which must be done manually using wmsetbg.

Some changes made using the WPrefs utility are immediate, while others require restarting Window Maker to take effect. Things like icon sizes, window auto-focus and window auto-raise all require restarting the window manager. The Exit**-->**Restart menu option in the Root Menu will restart Window Maker without requiring you to exit your X sessions.

Until I discovered how to resize the icons, I really hated Window Maker. I'm a minimalist at heart; my FVWM2 configuration uses very little screen space for the GoodStuff bar. The large icons are annoying to me, so I resize them down quite a bit. This has a drawback, however. Since clocks are docked appicons, they tend to require lots of space in order for me to see the clock analog hands or digital text. Similarly, many of the dock applications require at least a 64x64 appicon, or else you can't read their contents. In other words, they're cute, but they take up too much space for my taste.

One other thing about Window Maker is it supports sound. I don't like my computer making noises at me. That's why I have a stereo; I pipe CDs played on the computer through the stereo, so I don't use the sound features found in Window Maker or other window managers. However, sound support is there, if you want it.

Performance

As with my Enlightenment testing, I ran Window Maker on two systems: a 400MHz AMD K6 desktop with 256MB memory and a 200MHz Celeron laptop with 32MB memory. Both systems ran stock Red Hat 5.2 using glibc 2.0. I also ran with the latest release of Window Maker, namely the 0.61.1 release. Performance on the desktop was good in all cases. Performance on the laptop was nearly as good, although a little jumpy in a few places.

Window Maker is a little less resource-intensive than Enlightenment, mostly because Window Maker doesn't provide the wild window borders you can get with Enlightenment. That feature of E is visually stimulating, but also memory-intensive.

I found that Window Maker performed quite well on my laptop, certainly as good if not better than Enlightenment. Opaque window moves are the default and require constant redraws. This wasn't so noticeable on the desktop, but was obvious on the laptop. Opaque moves are fairly slow to update and make WM appear to be sluggish, but in reality it's not really bogged down by such movements. This can be turned off, forcing transparent (outline only) moves which give the appearance of much better performance.

Window shading was not very smooth on the laptop, even with the fast speed set. I found it worked much better if I turned off the animations on my laptop. When I double-clicked on the title bar, the window (except for the title bar) just disappeared. Double-click on the title bar again, and it comes back. Since this happens very quickly, it makes it appear as if the system is working more efficiently.

Animations, including 3-D animations for iconizing windows, were generally fairly smooth. Neither shading nor animations are as smooth as on my desktop, but neither is slow enough to be distracting or prevent me from doing other work. The bells and whistles of Window Maker can all be turned off, so performance shouldn't be a big issue for laptops.

Running Netscape and GIMP on my laptop with Window Maker was no problem. Of course, very large GIMP files will bog down any system, so I just worked on small web-sized images. Still, Window Maker ran quite well.

Documentation

There is actually a fair amount of documentation available for Window Maker. A user's guide for a previous release is on-line, but it's not quite up to date with the latest (0.61.1) release. A FAQ in text form comes with the source distribution, and an HTML version is available on-line from both the main Window Maker site (http://www.windowmaker.org/) and a few other user-supported sites. Man pages for all tools included in the source distribution are also available.

Summary

Window Maker has many nice features. For artists, Window Maker makes personalizing the desktop very simple. Themes are easy to create, easy to save and easy to install. This leaves the artist time to focus on the creative side—the creation of the tiles, backgrounds and other images used in the desktop. That's the way a desktop should work.

Window Maker also has many things I don't like. The icons are too big, initially. Since most Dock applications expect to be running in 64x64 icons, my laptop screen space is compromised. I can scale these down, but I lose the functionality in the dock appicons. I also don't like having the dock appicons running along the right side of my display. I like them along the top of the display, and I couldn't figure out how to change this orientation.

The lack of a visual pager is also a drawback. Although pagers take up screen space (and you'd think I would hate that), their usefulness far outweighs their

size. The Clip is helpful, but not as much as the pagers you can get with Enlightenment or FVWM2.

Most importantly, I never iconize anything under FVWM2, so all this icon twiddling in Window Maker is a bit annoying to me. I far prefer configurable menu bars, like FVWM2's GoodStuff, where I can use very small icons (mostly for visual appeal) but still have quick access to many different applications.

Despite its drawbacks, Window Maker is a solid performer in various environments. It supports both KDE and GNOME and has a very easy-to-use graphical configuration tool. Themes are a breeze to create and save.

The main web site for Window Maker is a good place to get started, but it lacks any real details. The man page is well-written, with lots of details on what directories are used and what they are used for. Most of the truly useful information you'll find on the Web will be at user sites.

Window Maker is available precompiled in most Linux distributions these days. You don't have to build from source, but it's fairly easy to do so. There are few external requirements to get it running. This is how Window Maker differs from E: you can skip the need to know about compiling and installing software from source, something that Enlightenment depends on in its current state.

I like Window Maker, but between it and Enlightenment, I still prefer the latter. Then again, I'm an experienced desktop user. If you're new to Linux, you may find the graphically configurable Window Maker a little easier to learn. Both provide multiple desktops, themed interfaces and graphic-based desktop management tools. It's mostly a matter of taste.

The last article in this series is supposed to be on AfterStep, a window manager very similar to Window Maker and also based on the GNUStep environment. However, I may go with blackbox or sawmill instead. Both are very stable and provide some minimalistic aspects I find interesting. AfterStep is very much like Window Maker, and I'd prefer to talk about a window manager with a different design intent. If you want to get a jump on me, start out over at http://themes.org/. You can find links to all these window managers there, as well as some very useful information on configuring and using your favorite.

Resources

email: mjhammel@graphics-muse.org

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

Builder Xcessory

Robert Hartley

Issue #74, June 2000

Builder Xcessory (BX) is a mature, high-end, object-oriented-based UNIX GUI development tool for Motif and Java, released just over a year ago for use on Linux.

- Manufacturer: Integrated Computer Solutions (ICS)
- E-mail: sales@ics.com
- URL: http://www.ics.com/
- Price: $250 US for personal-use license
- Reviewer: Robert Hartley

Builder Xcessory (BX) is a mature, high-end, object-oriented-based UNIX GUI development tool for Motif and Java, released just over a year ago for use on Linux. Along with Code Fusion from Cygnus, BXPro won *Linux Journal*'s “Editors Choice” Award for Best New Application: Software Development in 1999.

In addition to highly intuitive widget layout, BX allows the developer to:

- Build components by grouping widgets together and making them into primary classes, with their own methods and data interfaces, without the need to compile them into separate shared libraries for placement on the palette.
- Implement well organized ways to apply styles for a consistent look and feel. Many organizations such as government and military subcontractors have their own strict standards for how a GUI should look, declaring their own font, color and other conventions for particular systems.
- Transparently keep code intact while the GUI is undergoing evolutionary changes. Callback code is not lost when changes to the GUI are made.
- Invoke and use standard development tools, such as Code Fusion, Purify, CVS, SGI Developer Magic and others.

- Add new components and widgets on the palette and use them “live”.
- Develop C, C++, ViewKit, UIL and Java applications, using industry-standard Motif and Lesstif.

Introduction

Builder Xcessory is the finest X and Motif GUI development tool available today for OOD/OOP developers working in C/C++ and Java, with enterprise development tools in the works, including other GUI APIs. For application frameworks, it allows the use of ViewKit and a MotifApp-based C++ class system, in addition to straight C for those maintaining legacy applications.

ViewKit is ideal for C++ and Motif work, but for the platforms on which it is not currently available, the MotifApp-based C++ framework provided runs like a charm when used with either Motif or LessTif. I found this especially helpful while porting Motif applications to the Rebel NetWinder, since its ViewKit port is still in the works at the time of this writing. One advantage to using the free Lesstif libraries, when Motif is absent, is that source is compatible across all UNIX/Linux-based X11 platforms.

Running various window managers does not affect the operation of the builder, as it has the Motif libraries statically bound, which also means it can be run without having Motif installed at all.

License

The Linux version of BX does not mess around with licensing daemons, using an encrypted license string instead. As of this writing, BX is available on Linux for $250 US for the personal-use license.

On-line Documentation and Tutorials

In addition to the comprehensive hard-copy manuals, all documentation is installed locally in HTML format and is easily invoked from the help menu. The manuals also have complete tutorials, describing in detail the steps needed to make use of each feature of BX. Tutorial data files and example code abound, and updates are freely available via the ICS FTP and web site.

Distributed and Thin-Client Applications

As with most other GUI builders, it takes very little effort to implement what is referred to as the façade design pattern.

Modeling a complex object in a class, providing simplified access functions or methods to control it, and then using the GUI interface to view it allows us to make use of the Model View Controller (MVC) concept. This means we could, at

any time, convert our program to a CORBA-type distributed application. Simply by substituting a method invocation on a CORBA-based object from within our GUI's event response routine, referred to as callbacks in X parlance, we now have a modernized distributed application. Keeping our GUI only loosely coupled to the data means we have little or nothing to update when changing how the back end of our application is implemented.

Designing with BX

BX is a true WYSIWYG GUI builder. It will allow you to add widgets and components to other widgets and components intuitively. As you build up the GUI, the design tree reflects the widget hierarchy. This is more intuitive than other systems, where a widget is added onto the design tree and the widget placement appears on the screen. It is much more natural to interact with the widgets directly without this level of indirection. BX will also interactively show what happens to the window contents as it is being resized, making manipulation of Motif's more complex container widgets a breeze to configure.

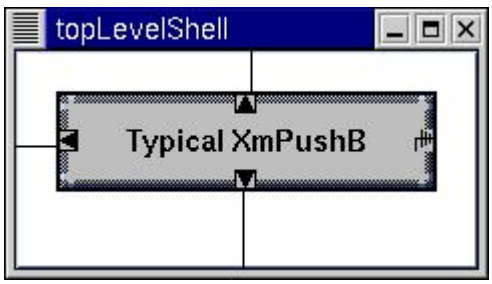


Figure 1. Illustration of widget constraint settings for a push button inside a form widget

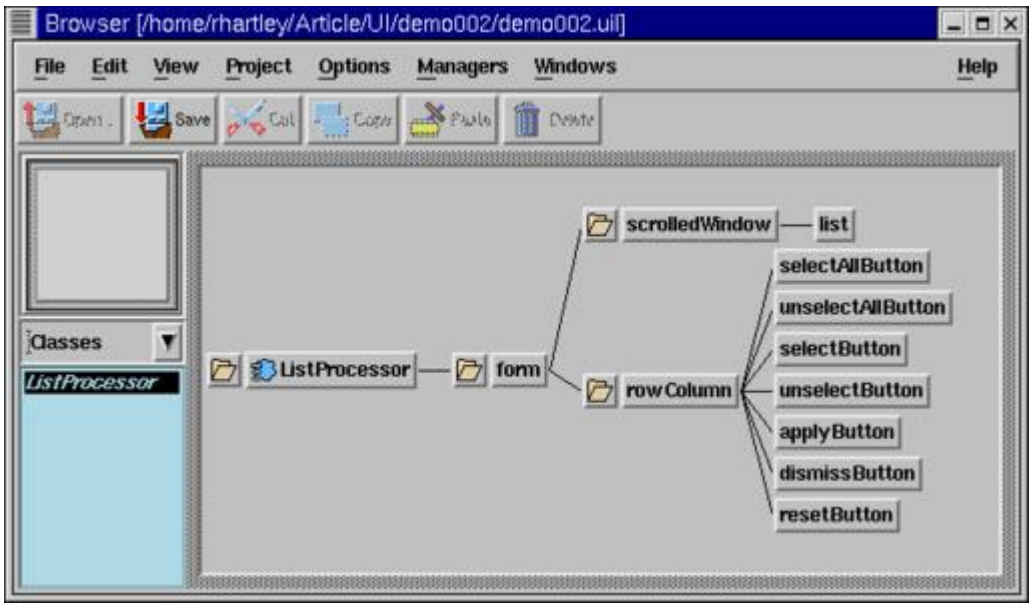


Figure 2. Browser window with sample widget tree displayed

Figure 3. Sample GUI as it can appear in the builder

Each Motif widget has its own list of resources which can be set using the resource editor. This allows for setting resources not related to geometry, such as fonts, colors and various callbacks.

There are a number of options for displaying which resources are viewable. As can be done with all of the BX menu items, the “View” menu shown in Figure 4 has been “torn off” and displayed in its own window, making for a more efficient workspace. If the user has selected the “All resources” option, there can be more resources potentially available for viewing than what is easily scanned through. BX provides a resource finder that allows the user to enter a search string, and any resource containing it will be scrolled to and highlighted, as shown in Figure 5.

Figure 4. View Menu

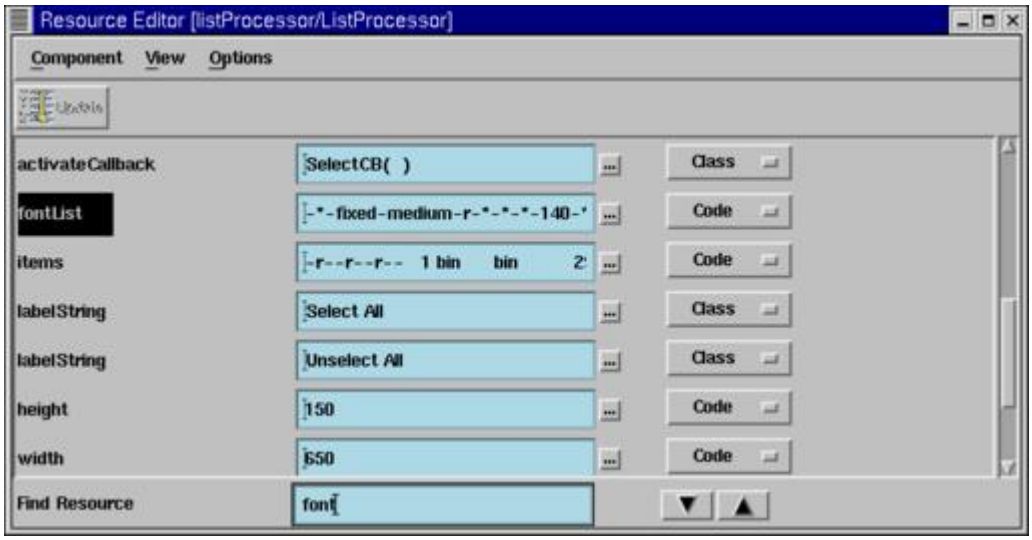


Figure 5. Search for Word “font”

Styles Management

The styles manager in BX allows us to apply pre-set values easily to the resources of any number of widgets. Suppose, for example, we wanted a pushbutton with a label indicating “Dismiss”, a bold font, with a black foreground and a somewhat olive-colored background; we could use the style editor as shown in Figure 6. Saving this style to a DismissButtonStyle, our style hierarchy tree would look like Figure 7. When we create a pushbutton and apply the style, it would look like Figure 8.

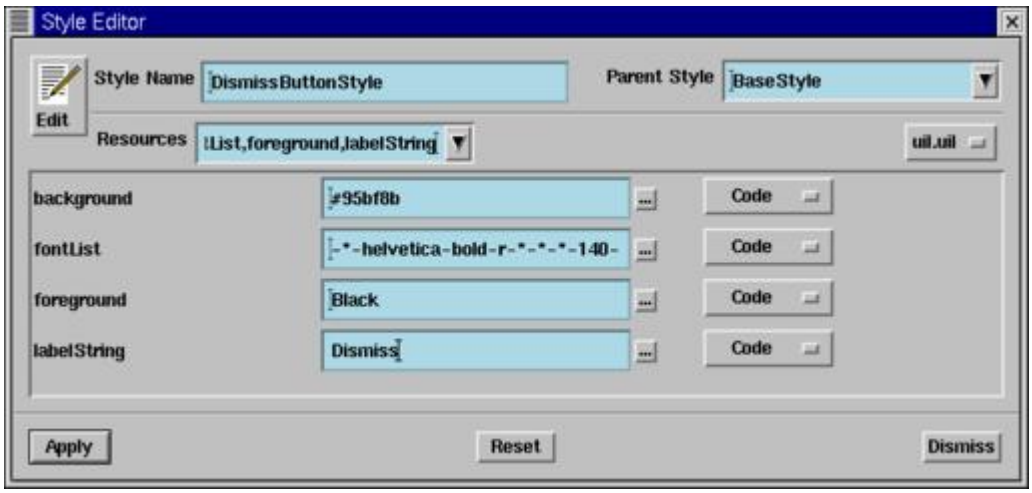


Figure 6. Style Editor

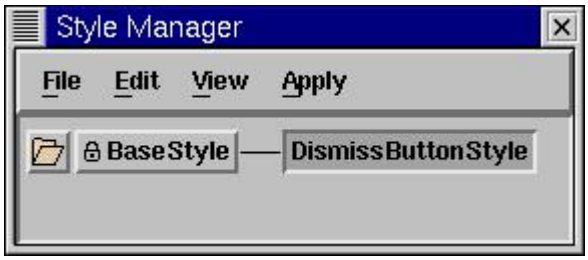


Figure 7. Hierarchy Tree

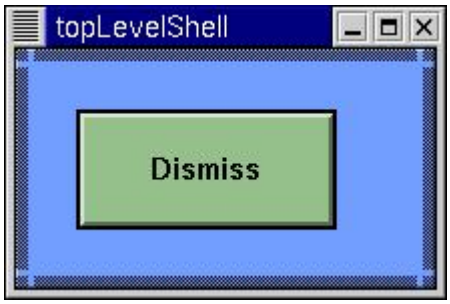


Figure 8. Push button in Defined Style

In addition to being able to apply styles to multiple widgets at once, styles can easily be made shareable so they are available to other applications being developed.

Creating New Classes

BX has two modes of looking at widget hierarchies: class and instance. Class mode indicates what the predefined appearance and behavior of a component will be, in effect defining new types. Instance mode tailors them for a particular widget or component, so we can tweak what makes them different from others of the same type.

From the most general class, we can define more specific sub-classes, each time progressively getting more refined or changing and overriding things as we go along.

To save time redesigning our standard layout, we can join a number of logically related widgets together into a single entity by turning it into a class. This makes reuse a snap, since all newly created classes get added to the palette.

To create a class containing one or more widgets, select the highest-level widget to use, then select the “Make Class” menu option. In the previous widget tree containing a form widget with a few buttons and a scrolled list, I performed the “Make Class” operation, and now have a new component called ListProcessor which I can treat as a stand-alone component. It is much more than a simple widget, because we avoid having to delve into the depths of Xlib, Xt and Motif, remaining at a relatively high level without ever giving up control of any of the underlying widgets.

In class mode, we can use the widget class tree to select which resources to expose, giving us a simplified means to control our newly created component without the clutter of having to view every resource of every widget it contains. As a convenience, any resources flagged as “exposed” have **set** type methods defined, so we don't have to worry about the exact Motif name for widget resources and the functions to set them.

Figure 9. Making New Class

New class methods (functions and procedures) and data can be added easily when we select the class entity in the browser. Our resource editor puts us into class edit mode automatically when we click the “New” button to add data or methods. (See Figures 9 and 10.)

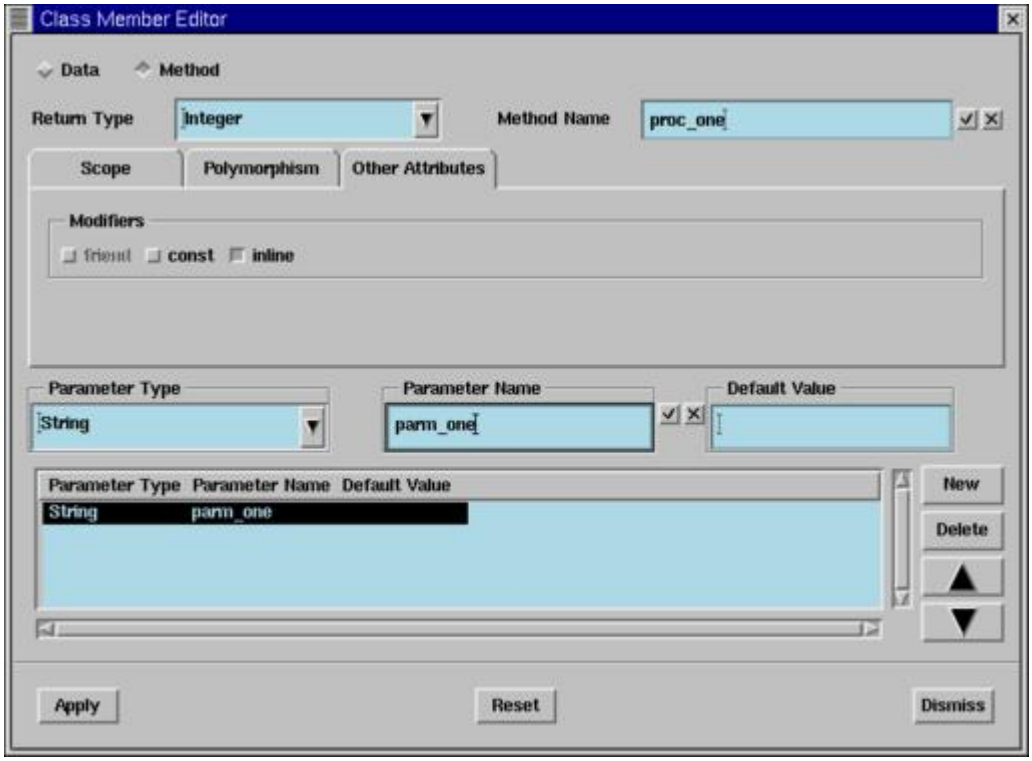


Figure 10. Adding New Class Methods

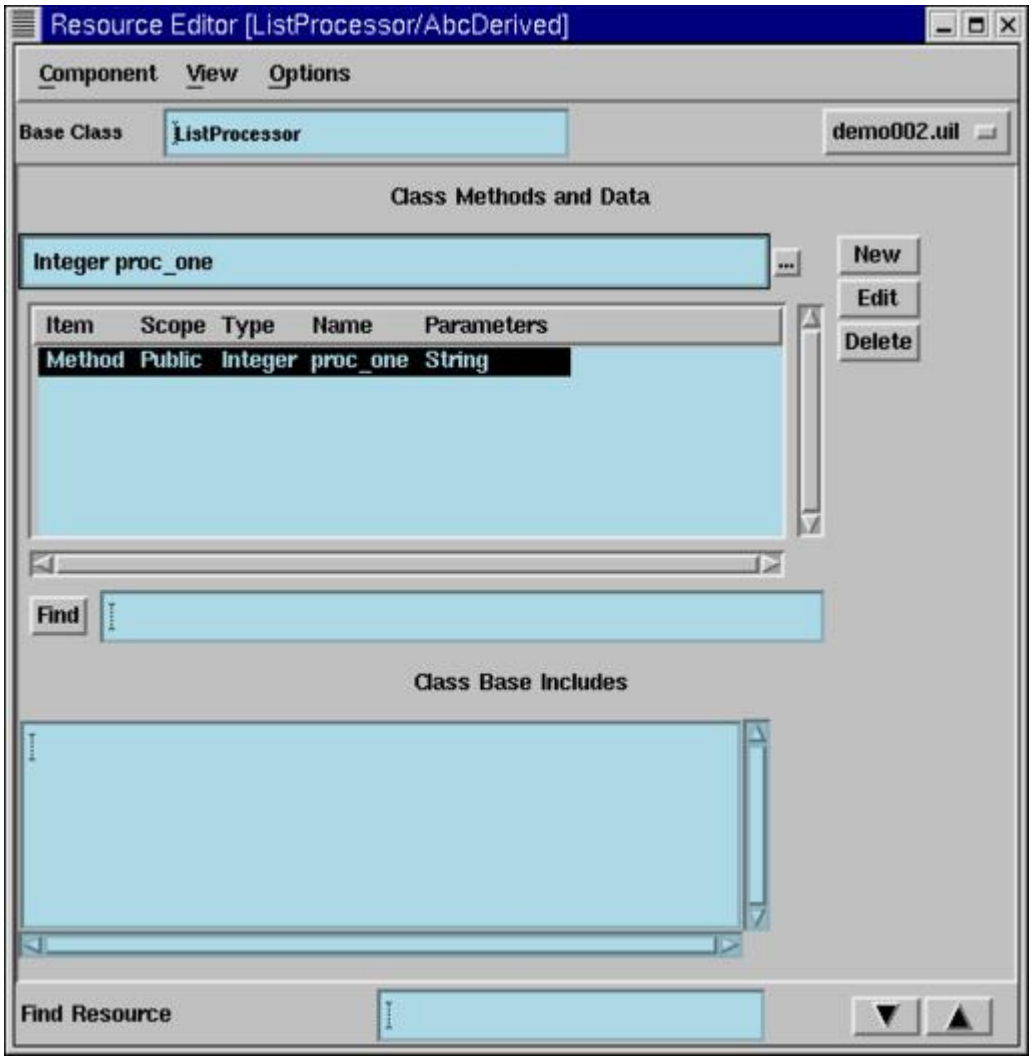


Figure 11. Resource Editor

Our resource editor now appears as shown in Figure 11, with the new entry inserted. We can now select the “Edit” button, and if needed, our source code will be generated before whatever source editor is brought up. At this point, we can look at our code with the cursor placed automatically on the line where our function starts.

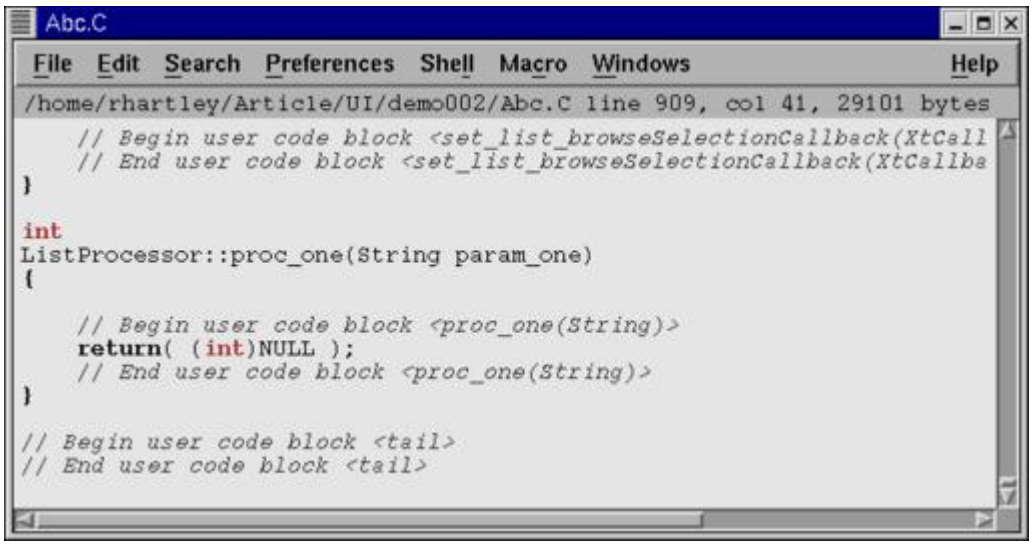


Figure 12. User Code Block

One thing we notice from the snapshot in Figure 12 are the “user code block” begin and end comments around the user code blocks. Anything between these can be modified by the programmer. BX reserves the right to make changes outside of these user code blocks, but will otherwise not interfere. BX avoids clobbering user-written code by scanning for and skipping over sections that are reserved by the developer. At the beginning of a project, these are empty lines bounded by the comment delimiters. Throughout the BX-generated code, these comment blocks will be found before, during and after any major code operation. This may seem superfluous to the new GUI programmer used to having only callback code stubs generated, but they can be a big help in tweaking final production code for major systems.

ViewKit and MotifApp

Until recently, Douglas A. Young was a principal scientist at Silicon Graphics Inc. (SGI). He has written a number of books on Motif development and is the most-quoted and read author on GUIs, having specialized on Motif in particular. His *Object-Oriented Programming with C++ and OSF/Motif* remains a much-cherished classic. In it, he develops the MotifApp GUI application frameworks, which have matured into the ViewKit product, originally available from SGI but now free on Linux, and commercially available for other UNIX platforms.

To understand what is meant by the “Doug Young method”, we need to look at the choices available to an application framework developer wanting to create an OOP framework using a GUI API based on a non-OOP language, such as Motif.

In some ways, Motif is an OO package. It uses inheritance to create new widget classes based on old ones, passing data and methods down from a super class to a newly derived subclass. Since it is written in C, some marshaling code is needed to overcome language features present in true OOP languages like C++. Gtk+ uses many of the same concepts, since it too is written in C, so the strategies here hold true for Gtk+ as well.

Doug Young's observations about using C++ and a C GUI can be paraphrased with his three strategies as follows:

- Our first option is not to use the OOP features of C++, simply using the compiler's stronger type checking. This, of course, will disallow us the greater advantages of using C++ and OOP.
- Our second choice is to encapsulate each and every widget into C++ wrapper classes. Although this may provide some aesthetic appeal, it

obfuscates the fact that adding extra marshaling code to obtain derived Motif objects is not true OOP, and can induce a performance hit compared to straight C. We cannot simply create a subclass of a C++-wrapped widget and expect it to work properly, since this would extend only the wrapper class and not the widget itself. In order to have an entire OOP widget class hierarchy, we need to have our widgets written in an OOP language from the ground up, the way the Qt widget set does.

- The third and most natural approach to using C++ with a C GUI API is to create high-level components composed of (instead of derived from) collections of GUI widgets. All programmers make use of this functional, instead of object, approach at one level or another, as we have yet to see an entirely OO CPU make it to widespread use. Even things such as the Java machines on a chip are, at some microcode level, non-object-oriented machines.

The ViewKit application frameworks, the commercial descendent of MotifApp, have been developed using this third approach.

To make things more intuitive, ViewKit has a feature called VkEZ, which is comprised of a header file and a library. Instead of forcing a programmer to use a number of Motif functions directly, VkEZ overloads many of the assignment operators, at the same time adding some extra helper functions. Examples of this include the ability to assign strings to text and label widgets directly and easily using the "=" operator. Although this may help get an initial version of a program up and running, it is usually the case that the VkEZ functions are replaced with their counterparts in the way of direct Motif function calls.

Summary

BX's maturity into a highly evolved, flexible tool has made it the tool of choice for serious OOP-based GUI development work for any Linux/UNIX-based platform. It is sure to remain in my software toolbox for years to come.

Robert Hartley can be reached via e-mail at rhartley@srougi.com.

Archive Index Issue Table of Contents

Advanced search

Visual SlickEdit 5.0

Larry Ayers

Issue #74, June 2000

Customization possibilities are extensive.

- Manufacturer: MicroEdge, Inc.
- E-mail: sales@slickedit.com
- URL: http://www.slickedit.com/
- Price: $295 US
- Reviewer: Larry Ayers

I can't think of many commercial software packages that have been continuously evolving since the early days of the personal computer industry. Visual SlickEdit, MicroEdge's award-winning programmer's editor-cum-development environment, was initially released in 1988 as a DOS and OS/2 character-mode editor. This was when IBM was marketing text-mode-only OS/2 as “a better DOS than DOS”.

Clark Maurer, currently CTO of MicroEdge and still active in SlickEdit development, was employed at IBM's Watson research lab. He was one of the developers of the legendary internal IBM editor E; this experience enabled him to quit IBM and begin development of the first SlickEdit releases.

Installation

Having installed an earlier version of SlickEdit from floppy disks a couple of years ago, I anticipated no problems installing version 5.0 from CD-ROM. A GUI installation interface makes the process even easier than before, but I ended up with an installed editor which wouldn't even start up. After an exchange of e-mail and phone calls with MicroEdge's responsive and helpful service personnel, I still couldn't get it to work. I e-mailed a core-dump file to MicroEdge; it seems there were some compatibility problems with the particular version of Debian I've been running, which admittedly is the

“unstable” release. The support staff even went so far as to install Debian 2.0 on one of their test-bed machines, and reported that SlickEdit started up just fine. Since there didn't seem to be any immediate solution to this problem, I installed SuSE Linux on a spare partition and soon had a functional editor. I imagine this particular incompatibility hadn't come to light before due to the nature of SlickEdit's normal user base, which probably uses older and more proven versions of Linux.

Customization possibilities are extensive. Aside from the expected keymap and syntax-highlighting adaptations which can be made, menus and pop-up dialogs can be altered and even created from scratch. Much of the editor's functionality is written in a C-like language called Slick-C, but the language itself doesn't need to be learned, as SlickEdit comes with a Slick-C IDE (integrated development environment) called the Form Editor. With this editor (which resembles Visual C), existing dialog boxes can be modified and new ones created from scratch. These two screen shots illustrate the HTML font dialog box opened for editing.

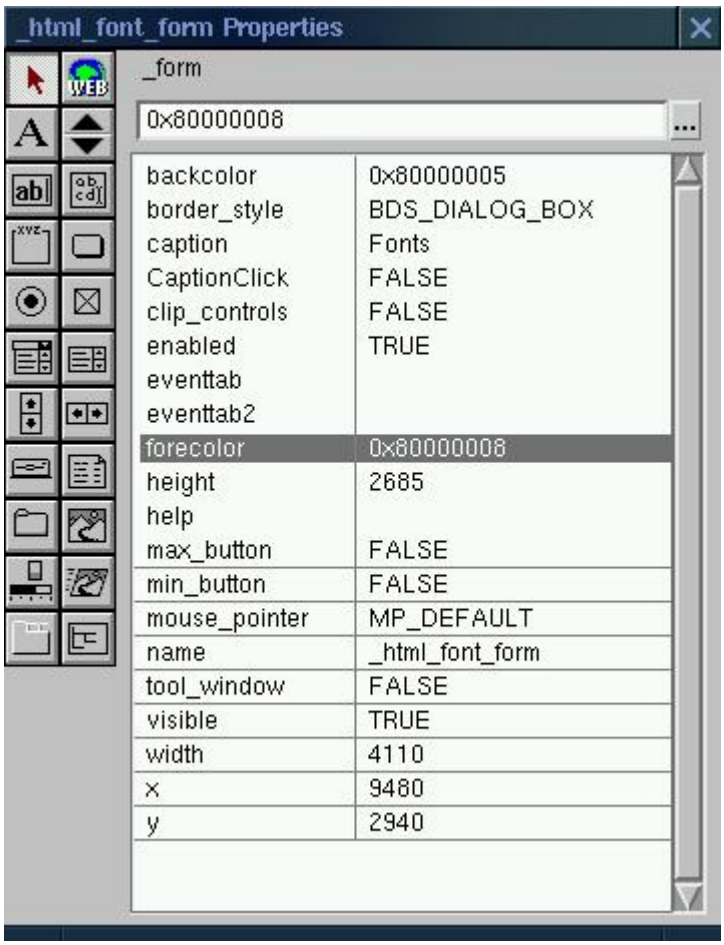


Figure 1.

Making the guts of a program accessible to users, which enables almost unlimited tailoring of a program to individual (or corporate) preferences, is a familiar concept to Emacs users, but rare in commercial products.

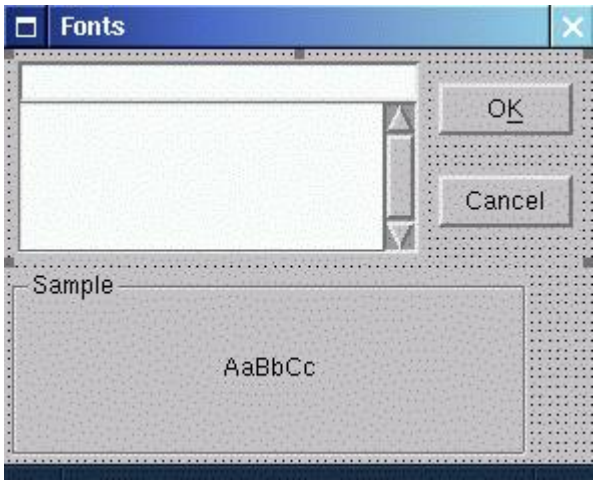


Figure 2.

Features

SlickEdit is feature rich. Leaving aside the basic abilities any good programmer's editor should have, this editor abounds with useful functions, including:

- SlickEdit is unparalleled in its broad base of supported platforms. Along with Linux, versions are available for Solaris, HP-UX, Digital UNIX, Windows 95/98/NT, SGI-Irix and IBM OS/390. The look and feel is identical across all platforms.
- An extensive array of programming languages are supported, while new languages can be added easily.
- Context Tagging can be quite useful. Expression type, scope and inheritance analysis are performed as you type. Members and inherited members of objects are displayed in a dialog, while at any time, you can navigate with a keystroke to the definition of an identifier, including class members, functions and variables.
- Sophisticated file differencing (called DiffZilla) is similar to the traditional Linux diff utilities, but is controlled via dialog boxes and is thus easier to learn and use. The Diff dialog can operate on multiple files in a directory tree, while other directories can be excluded from the operation.
- The search and replace functions are analogous to those in Emacs, even to the extent of including the invaluable incremental search, which allows searches to be completed as a phrase or string is typed.
- Up to fifteen clipboards are available. Their contents are accessible from a dialog box and are saved between sessions.
- Window treatment is easily controlled from the menu bar, especially important in an editor with so many possible windows. Tiling, cascading and linking windows are among the possibilities. All windows are confined within the main SlickEdit window and can be positioned and resized just as any X window can be.

- Code beautification, which reformats indentation, tag styles and brace styles to your preference, is helpful when viewing or editing someone else's files. Version 5 of SlickEdit not only lets you beautify C, C++ and Java files; now, JavaScript and HTML (and HTML with embedded JavaScript) are also supported.
- SmartPaste is another editing convenience. Paste a cut or copied block of code into a file, and it will be re-indented automatically to the appropriate level.
- SlickEdit makes good use of the IDE-inspired concept of projects, assemblages of source files and compiler command lines along with the working directory. These can be loaded as a unit and shared with other programmers. Another useful concept is the workspace, a higher-level collection of projects. Dependencies can be defined within projects in a workspace, so that one project must be built before another. Projects can inhabit more than one workspace, and (as with projects) workspaces can be shared with colleagues. Visual SlickEdit can open workspaces and projects created with Visual C++ and Tornado.
- When more than one programmer is opening and modifying files and projects, some form of version control is essential. SlickEdit has built-in support for commercial version-control systems such as ClearCase, SourceSafe, SCCS and several others. Adding support for a new version-control system can be done easily.
- This release sports enhanced HTML-editing capabilities. When used with the extensive collaborative and distributed development features, SlickEdit has positioned itself as a viable platform for web content creation.
- Spell checking is exceptionally versatile in this release. Checking can be limited to comments and strings in source files, while HTML spell checking automatically ignores tags. Multiple files can be checked with one command, even recursing down into subdirectories.
- Although it may seem odd for an editor to double as an FTP client, this feature can, in effect, let a user edit files on a remote machine as if they are local. Rather than manually downloading, editing, then re-uploading files to a site, the built-in FTP client lets these remote files appear and be accessed as if they are in just another local directory.
- Previous versions of SlickEdit allowed users to emulate Emacs, Brief and vi key bindings, enabling UNIX users to come up to speed quickly. Version 5 introduces Visual C++ emulation, providing a similar service to users in the Windows camp.

Tagging and Tag Files

Tagging of source and header files is an important part of SlickEdit's functionality. Emacs and vi users will be familiar with **ctags** and **etags**, two separate utilities for generating tag files. Tagging is useful in any programming project with more than a few source files, as it helps the programmer keep track of just where functions and classes are defined. Usage of tag files in SlickEdit is similar to that in Emacs; a keystroke summons a window with the tag reference. SlickEdit's approach is easier to learn for those unfamiliar with Emacs, vi and UNIX in general. As with other UNIX-like functionality in SlickEdit, instead of using separate command-line utilities with their own peculiar syntaxes, etc., the tagging utilities are built into the editor and are accessed via dialog boxes.

The first time a new user starts up SlickEdit, the tag-generation utility attempts to build a tag-file database of source-code files in the current directory, if any, and of general compiler header files. In my case, the header files in /usr/X11R6/include and /usr/include, as well as the Linux kernel source header files, were all catalogued. Tag files are used by the Class Browser, allowing the user to quickly find classes, functions, variables and other code constructs referred to in source files.

SlickEdit's tagging facility is an impressive piece of work, effectively bringing an esoteric and somewhat abstruse but enormously useful facility to a broader programming community.

Documentation

Bucking the lamentable software industry trend toward less documentation, SlickEdit is bundled with a three-hundred-page hard-copy manual, and the same material is also available on-line through an integrated help facility. This documentation is clearly written and well-illustrated. A complete reference to the Slick-C language is included, a helpful feature for project leaders with a desire to tailor the editor to the work at hand.

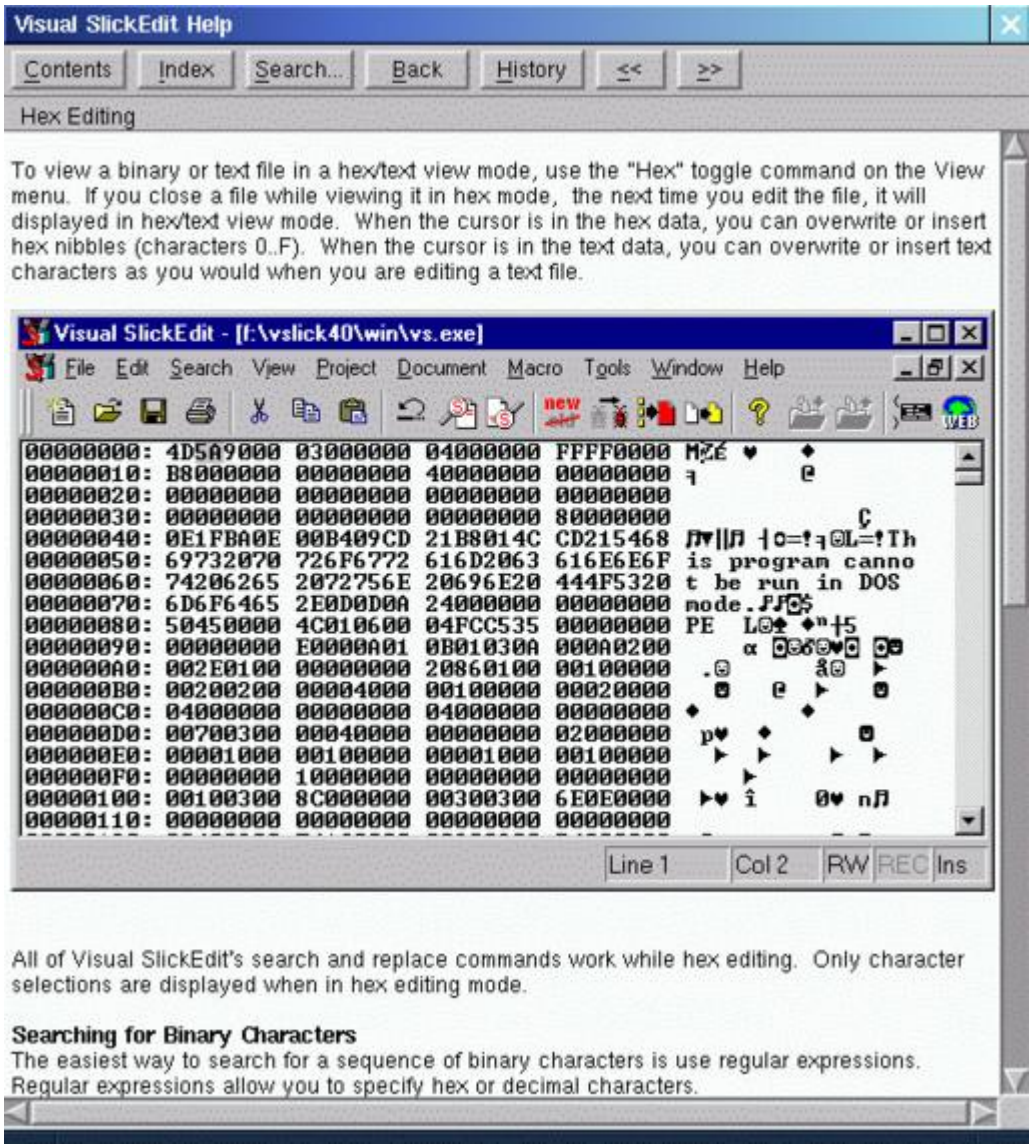


Figure 3.

Free vs. Commercial

At $295.00 per copy ($99.00 for an upgrade from a previous version), prospective users might wonder if the program is worth that much, considering so many free, high-quality editors are available for Linux, each with active and responsive developer and user communities. The answer is: it all depends on how much time you have available for learning to use an editor effectively, and even more important, how much time and inclination your co-workers have to spare. Nearly all of the functionality of SlickEdit can be duplicated by the various versions of Emacs or vi, but only at the expense of time for tracking down the necessary packages and configurations. This may be second nature for programmers imbued with the UNIX philosophy, but what about that co-worker who is more accustomed to Visual C++? SlickEdit provides a sort of neutral ground, an environment which is acceptable to programmers from a variety of backgrounds. The cross-platform nature of this editor is one of its strongest selling points, especially significant now that more and more

commercial software is being released in versions for a variety of operating systems.

SlickEdit probably won't appeal to many solitary or academic programmers; its niche is corporate programming environments, where a common style of code and project files can increase productivity.

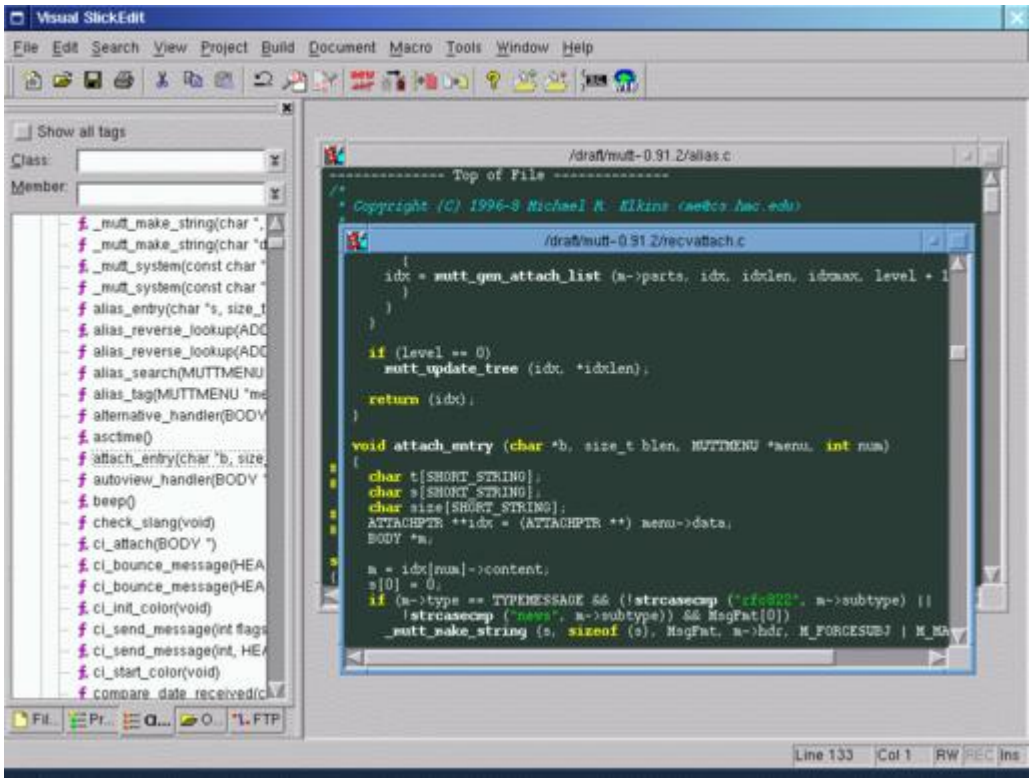


Figure 4.

Availability

MicroEdge maintains a web site for SlickEdit users and prospective customers at http://www.slickedit.com/. Patches for registered users are available there, as well as macros and tips from other users and the MicroEdge developers.

Conclusion

SlickEdit is a mature Linux product; this isn't one of those Windows applications quickly ported to Linux. MicroEdge has been supporting Linux for several years now, and this is evident in the stability and usability of the editor. While SlickEdit may be priced out of the range of the typical Linux home user, the price is not out of line for a product intended for business use. The advantages for group usage, including sharing files, projects and configurations, along with excellent support from MicroEdge, make the product a good value in its intended market.

The Goods

- Cross-platform
- Excellent documentation
- Extensive customization
- Feature rich

The Bads

- price will not appeal to home user

Larry Ayers lives on a small farm in northern Missouri, where he raises sheep, shiitake and shell scripts. Between bouts of running a portable bandsaw mill and helping local people and businesses with computer problems, he tries to find time to play the violin and cello. His e-mail address is layers@marktwain.net.

Archive Index Issue Table of Contents

Advanced search

Photodex's CompuPic

Michael J. Hammel

Issue #74, June 2000

There is much more to graphics on the Linux desktop than games, 3-D modelers and renderers.

- Manufacturer: Photodex
- E-mail: info@photodex.com
- URL: http://www.photodex.com/
- Price: free for non-commercial use
- Reviewer: Michael J. Hammel

I've been following graphics tools on Linux now for a number of years. Graphics tools are the domain of the desktop user and, as such, have been slower in coming from both the open-source and corporate worlds than networking and server-side tools. Fortunately, this is a trend that's starting to change. The importance of Linux on the desktop was recognized by video-card companies with the advent of video games from Loki and high-end tools like Houdini from Side Effects Software. But there is much more to graphics on the Linux desktop than games, 3-D modelers and renderers.

One company that recognized early on the importance of Linux for graphics users is Photodex, a software development company focusing on graphics and digital-content management. Photodex, located in the heart of Texas' high-tech mecca, Austin, started porting work for their CompuPic product in early 1999, with a first public beta release in June of that year. Recently I was able to sit down with a registered version of CompuPic 4.6 to find out just what this product can do.

CompuPic is an X Window System-based digital-content management tool, which in layman's terms means it's designed to assist users in keeping track of graphical images, animations and sound files on disks and networks. Although tools like this are most useful to web developers and graphic artists, the

average Linux user will also find significant value in products of this nature. Available for Windows, Macintosh and Solaris, CompuPic is currently available for only Intel x86 Linux, although support for Alpha and PowerPC users may be available in the future.

Once started, CompuPic opens a four-part window. The top part contains a traditional text menu system with an icon-based toolbar directly underneath. Below these are the Folder List and File List regions. Below the Folder List is a Preview Window. The Folder List looks and behaves much like File Explorer from Windows. The list starts from the root directory (i.e., "/") and each directory is given a file folder icon in either the open or closed form. Opening a directory by clicking on it once will show all files and subdirectories inside.

The File List is a list of individual image, sound and animation files. This list can take a number of forms, from simple thumbnail/file name combinations to complex thumbnail, file name, type and size listings. These variations, along with multiple sort options, make finding files rather simple. What's truly interesting is that, although designed specifically for image, sound and animation files, the file lists and sorting options work with any file type. CompuPic is a powerful file management system no matter what type of files you happen to be working with.

CompuPic works extensively with thumbnails. The File List window has a number of user-configurable sizes for its listings. The Preview window also shows a thumbnail of full-size images, providing a more detailed view of the image file (if it is an image file, otherwise nothing is shown in the preview). Double-clicking on an entry in the File List window, or clicking once on the Preview, will give a full-screen display of the image. User preferences allow you to set the scaling used for the full-screen display. It's also possible to select multiple files from the File List.

Two important features of CompuPic, Slide Show and MaxiShow, make use of the full-screen display to show one or more selected images. The Slide Show will display images in the order selected from the File List, putting a user-configured delay between images. The MaxiShow is similar, except it can display multiple rows and columns of images, also with a user-configured delay. While in full-screen mode, you can get a menu of image-management options by moving the mouse to the top of the screen. Options include a small set of image manipulation tools (blur, sharpen, brighten, darken), image transforms (rotate, scale and so forth) and a few special features like adding balloon comments to images.

While playing with the MaxiShow feature, I discovered one of the most interesting features about CompuPic—it actually makes use of the **PAUSE** key.

I've never seen an application use that key, after 20+ years of software development and computer use. Even more interesting, it actually causes the application to pause. You can use this incredibly intuitive (but fairly unexpected) feature to stop and restart a slide show.

CompuPic is full of very useful features. Unless you work for print publications, the film industry or somewhere else in the graphic arts industry, you might not think there are many things to do with graphic images. One handy feature is the quick picture indices, where a set of images is placed as thumbnails on a single page for printing. These are generally referred to as contact sheets by those in the business, but who cares what you call them? How incredibly handy it would be to have printed indices of your on-line photos from this year's family trip to Lake Whatchamacallit! You can then use another feature—e-mailing images and indices—to send the contact sheet to family members and let them pick the images they want. No more getting ten copies of entire rolls of film just so you can send a duplicate to everyone of that one picture of Uncle Ernie falling into the lake. If everyone likes it, Uncle Ernie's fame can live on in an electronic greeting card, which you format and e-mail right from CompuPic.

A recent addition to the beta and public releases of CompuPic are contracts with various on-line photo communities. Photodex includes options for connecting to four such communities: PhotoLoft.com, ofoto.com, PhotoIsland.com and PhotoPoint.com. The connection is weak—CompuPic will attempt to connect to the web site (a function that failed to do anything more than open a Netscape window on my box), but there didn't seem to be any way to do the uploads directly from CompuPic. In any case, each community offers limited web storage for your images, options for making greeting cards from those images and formatting the images into personal photo albums. Some of the sites also offer related articles, such as digital camera and scanner reviews. The value of such on-line sites is, of course, purely subjective.

Good Design, But Stability Beckons

CompuPic is well-designed in a number of ways. First, the database of images is kept in a private directory in the user's home directory. This means users need not worry about making unwanted changes to an image directory on a web server. The database is relatively small, even for large images or a large number of them. Another good design is the interface—it's quite intuitive. Once you're familiar with the basic layout, it's easy to find more features. Changing file names, for example, required only clicking on the file name of the selected entry in the File List, then typing right over the old name.

Photodex offers some application-specific security within CompuPic by allowing users to provide user IDs and passwords for a given copy of the program. However, such protection isn't that helpful, since the actual files are still

potentially accessible using normal Linux file access privileges. Photodex recommends proper file-system administration for true file security.

One design feature which deserves special recognition is the terrific Help system. It's a hyperlink-based system that includes plenty of images to accompany a fairly thorough and well-indexed text. The Help system is complete enough to make using CompuPic possible without printed documentation—something I can't say is true for many commercial applications.

Although the overall design and feature set is quite good, CompuPic has a number of problems. A minor one is how it requires you to be connected to the Internet in order to e-mail an image, even if you're e-mailing it to a user on the local system. I don't know why this is, but trying to send an image to myself without being connected to the Internet failed. The test was to send the mail to my local user ID without specifying a domain name. This should have been routed locally, which is what my normal Sendmail configuration does.

However, this is a modest problem. CompuPic has much larger problems—stability, for example. I ran CompuPic on two systems: a desktop box running Red Hat 5.2 with 256MB memory and the Xi Graphics Accelerated-X X server, and an IBM ThinkPad 1410 with 32MB memory and an XFree86 X server. I had numerous crashes on both boxes, although it was worse on the laptop. Photodex states the program had not been tested with Xi's X server. There were a few display problems with this server in the File List window—menus posted over this region were not always cleared completely. This didn't happen with the XFree86 server. Display problems were minor. The biggest problem was crashing.

On the laptop, I couldn't change directories in the Folder List without the program crashing unless I tried compacting the database first, and that let me change directories only once. This problem never showed up on the desktop system. Both systems had problems with rendering multi-line text in the Greeting Card feature, and the laptop version wasn't happy with changing virtual terminals. Upon returning to the X session, I had to hit the **ENTER** key to get CompuPic to respond. If I didn't do this, I couldn't do anything in the X session—CompuPic had taken keyboard and mouse focus to wait for that one key press. Photodex had most of these problems listed on their bug-tracking web page. One bug listed on that page was a crash at startup on Red Hat 5.2 systems. It appears there may be some stability problems for CompuPic when run on glibc 2.0 systems.

Other annoying issues include the About and Print options doing nothing. I wasn't able to print anything. I suspect this feature wasn't complete in the

version I had (Version 5.0 Build 1032), which is still a beta release. The Help text, while an extremely useful feature, assumes a Windows or Mac platform in some places. This is an extremely minor issue, but attention to detail is what separates good programs from great ones.

Although the Help system talks about support for scanners and digital cameras, the associated menu option is missing in the Linux version. This is something PhotoDex will want to look into, as it's become a big issue for many Linux users.

Summary

CompuPic is offered in a free version to end users, while corporations need to register their copies. Red Hat was suitably impressed—they will be including the product with their next release. Regular users can download a copy from http://linux.compupic.photodex.com/.

As I told Jason Cohen at Photodex, I found the longer I used CompuPic, the stronger my love/hate relationship with it became. It has many very useful features, whether you are a home hobbyist or a professional photographer, and upcoming features, such as the macro-recording feature, will make it even more impressive. But the beta versions have many problems. Stability is an important issue that needs to be addressed. Crashes abound, though never were the images or any other files corrupted. Still, despite the numerous crashes, I found I could perform quite a bit of work with CompuPic. That's a pretty good start for PhotoDex.

The Goods

- Well-designed
- Good feature set
- Quick picture indices
- Ability to e-mail images and images
- On-line photo communities

The Bads

- Beta version
- Stability problems (frequent crashing)
- Requires Internet connection for E-mail
- About and Print options don't work
- Missing menu items

Michael J. Hammel (mjhammel@graphics-muse.org) is a graphic artist wannabe, 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

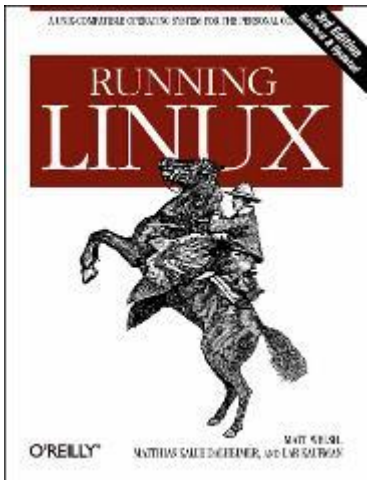
Advanced search

Running Linux, 3rd Edition

Ibrahim F. Haddad

Issue #74, June 2000

This new third edition surpasses all the old ones in terms of the topics covered and the extent to which it explains the material.



- Authors: Matt Welsh, Matthias Kalle Dalheimer, Lar Kaufman
- Publisher: O'Reilly & Associates
- Price: $34.95 US
- ISBN: 156592469X
- Reviewer: Ibrahim F. Haddad

Running Linux is very well known in the Linux community as one of the best getting-started books for Linux. It explains everything you need to understand, install and use the Linux operating system. It includes a comprehensive installation tutorial, complete information on system maintenance, tools for document development and programming, and guidelines for network, file, printer and web site administration.

I received complimentary copies of the first and second editions. However, this new third edition surpasses all the old ones in terms of the topics covered and the extent to which it explains the material. It features several new topics such

as KDE, Samba and PPP, as well as revised instructions for installation and configuration, particularly the Red Hat and SuSE distributions.

The book's first three chapters are a gentle, very well-organised introduction that covers the history of Linux, its system and software features. It then goes into the preparations involved in installing Linux, such as hardware requirements, different kinds of distributions and post-installation procedures.

In Chapter 4, readers are introduced to the basics of UNIX commands and concepts. These cover logging in, setting a password, virtual consoles, popular commands and shells. Then it moves into file name expansion, manual pages, file ownership and permissions, processes, startup files and the directory structure.

Chapter 5 goes into system management issues such as booting and running the system, the /proc file system and managing user accounts. Managing file systems, swap space and devices are covered in Chapter 6.

Earlier editions of *Running Linux* served as central guides on installing, configuring and using the OS. A discussion of upgrading the software and the kernel starts in Chapter 7. It covers archive and compression utilities, using RPM, building a new kernel, loadable device drivers and loading modules automatically. I found this part very technical, concise and to the point.

Chapter 8 covers administrative tasks used in maintaining the system such as making backups, scheduling jobs using **cron**, managing system logs, managing the print services, setting terminal attributes and what action to take in an emergency situation.

Chapter 9 is the Editors and Text Tools niche. It shows users how to edit files with vi and Emacs and how to print documents and use different graphics programs.

It's not until Chapter 10 that the authors cover installation and configuration of the X Window System. This chapter covers X concepts, its hardware requirements and how to overcome some of the problems you may encounter when you install it. A natural followup to Chapter 10 is “Customizing the X Environment” in Chapter 11. This includes the basics of X customization, the FVWM window manager, the K desktop environment and X applications.

Chapter 12 discusses Windows compatibility and Samba, with a lot of details on sharing files and programs.

Chapters 13 and 14 cover programming languages and the tools used by programmers. It includes **gcc** (the C/C++ compiler), Makefiles, Perl, shell scripts, Tcl/Tk, **gdb** (the debugger), profilers and other programming tools.

In Chapter 15, the authors explain networking with TCP/IP and PPP (dial-up PPP and PPP over ISDN). It shows you how to set up your machine to connect to a network or to an ISP. It also covers NFS and NIS configuration. This section is particularly clear and readable.

The last chapter is about the World Wide Web and electronic mail and shows how to configure e-mail, set up the Elm and Netscape mail readers and even how to run your own web server.

The book comes with eight appendices. These are of a good value, since they cover where to get more information on Linux, the GNOME Project and installing Linux on different platforms. In addition to that, there is a whole appendix on LILO boot options, which I found very useful, and another one on Zmodem File Transfer.

The book's style is informative, clean and to the point. The language is concise and easy to read. There are several well-crafted sections, such as rebuilding the kernel and configuring network links and servers.

Running Linux is a great introduction to the world of Linux. Some people go further than that in considering it the Linux Bible. I believe it is a good reference and by far the best “introduction to Linux” book. The only drawback I could think of is the missing Linux CD-ROM. If it came with a Linux distribution, this would make it the most complete package for first-time users.

All said, I would definitely recommend this text to beginners and intermediate users. It will be a good addition to your Linux library.

Ibrahim F. Haddad (ifhadda@cs.concordia.ca) is a systems analyst at Concordia University (Montréal, Canada). He has been using Linux since 0.99. Ibrahim is currently in the Computer Science doctorate program at Concordia. His research interests are distributed object technology for Internet computing, e-commerce and web-server load balancing.

Programming the Perl DBI

Bill Cunningham

Issue #74, June 2000

This new book from O'Reilly is the first one available from any publisher on the subject of programming the Perl DBI, and its authors are imminently qualified to write it.

- Author: Alligator Descartes and Tim Bunce
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $34.95 US
- ISBN: 1565926994
- Reviewer: Bill Cunningham

This new book from O'Reilly is the first one available from any publisher on the subject of programming the Perl DBI, and its authors are imminently qualified to write it. Tim Bunce wrote the DBI Perl module, and also the DBD Oracle module. Alligator Descartes, who originally began working on the book, is a leading Perl/DBI exponent in the Oracle community. The authors have been

working on the Perl interface to relational databases, and documentation thereof, since 1992. *Programming the Perl DBI* is not merely a republication of older documentation. A great deal of new material is here, not all of it about DBI.

The book starts out very simply, describing how even a flat file like /etc/passwd can be a bona fide database. The authors progress logically to more complex data models by addressing the shortcomings of the simpler models. After flat files, the book covers DBM and the Berkeley Database Manager, again noting the details of use, pros, cons and applications where this type of database is best used. A chapter on SQL and relational databases is included as a primer for those who are new to the material.

Three chapters deal with the theory and practice of using DBI to interact with various relational database engines. One chapter is devoted to ODBC. Every DBI method is described in detail and illustrated with at least one code example. There is also a chapter on DBISH, the DBI shell, wherein one can interactively give commands to the database after the manner of Oracle's SQLPlus. This tool will be valuable during development of database programs.

The book includes a description of a complex relational model where Microsoft clients access UNIX servers and vice versa over a network, with data encrypted via software.

The appendices contain a complete DBI specification, and specific information for each of the major relational engines.

The authors assume the readers have significant fluency with Perl, including installation and use of Perl modules and Perl's object-oriented syntax. Although the code examples are short, numerous and well-documented, readers new to Perl may need a Perl reference as they go through the code.

As a one-source database reference, the book is essentially complete. Detailed treatment is given to the DBI/DBD concept where each database engine has a unique driver (DBD), and a DBI superlayer that provides transparency to the engine below. All of the DBI methods are covered in detail, so the reader is able to select methods appropriate to his or her programming tasks. The authors are especially thorough in their treatment of debugging levels, error trapping and error interpretation.

They state that DBI is very much a work in progress and envision it to be a full-featured product soon, on par with any commercial development product available.

I noticed only two minor typos in the entire book. This was quite refreshing to me, as I have read computer books containing so many errors as to be virtually useless.

Anyone who works with data will find this book useful. And the more closely one works with relational databases and the programs which access that data, the more useful it will be.

Master Sergeant **Bill Cunningham** , USMC (bwc@coastalnet.com) is a systems administrator at Cherry Point, North Carolina. In addition to Linux and Perl, he is into model rocketry, Cub Scouts, t-ball, and lots of other stuff.

Archive Index Issue Table of Contents

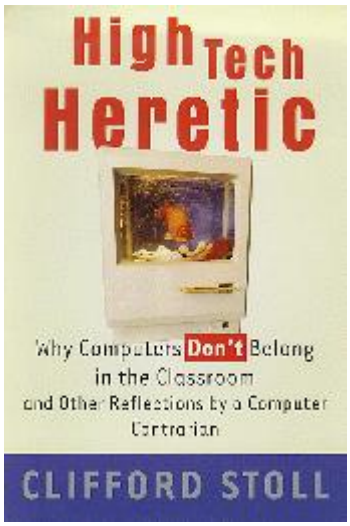
Advanced search

High Tech Heretic

James Paul Holloway

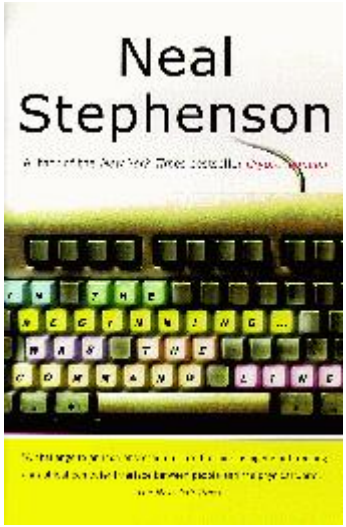
Issue #74, June 2000

It is becoming increasingly important to consciously debate and consciously control the effect of computing on culture.



- Author: Clifford Stoll
- Publisher: Doubleday
- ISBN: 0385489757
- URL: http://www.randomhouse.com/
- Price: $24.95 US

In the Beginning Was the Command Line



- Author: Neal Stephenson
- Publisher: Avon Books
- ISBN: 0380815931
- URL: http://avonbooks.com/
- Price: $10.00
- Reviewer: James Paul Holloway

Those with high-tech products to sell, no matter if their vision is cynical or genuine, are able to exercise a tremendous influence over our society. They have convinced us that broken software is to be expected, and a fix will be out soon; that a new computer every few years is necessary; and that children need to use computers practically from infancy. And their influence is not limited to consumer-users. Because of the conventional wisdom that “high-tech” is responsible for the current economic boom, they also have a strong influence on political, educational and legal figures, for example, selling the notion that copyright of software is a special legal form requiring special criminal provisions not previously needed, or pushing for the adoption of multimedia-based educational tools of uncertain educational value.

It is therefore becoming increasingly important to consciously debate and consciously control the effect of computing on culture. Fortunately, there is a growing literature of computer contrarian books that argue against the conventional wisdom of computer software design and deployment. The literature I am writing of is not about software engineering, but rather the purposes to which software is put and the way it is developed for human use. Still, it is amazing that none of these books are written by the key open-source players. Perhaps the open-source activists are too busy building the tools of the revolution to worry about how those tools are being used.

Two recent examples of computer contrarian books are *High Tech Heretic* by Clifford Stoll and *In the Beginning Was the Command Line* by Neal Stephenson. These are notable because they were both written by experienced authors who love and are experienced in computing, and whose previous successes in techno-book writing has made them somewhat famous among members of the hacker community. Both authors set out to challenge dominant ideas in our current culture of computing, using dramatic images and emotional rhetoric rather than through a data-driven discussion.

In *High Tech Heretic*, Stoll argues against the whole notion of ubiquitous computing, against the idea that computers not only can but should cause fundamental changes in the way we do everything. The misguided deployment of computers and Internet connections in schools earn his special ire, as these technologies may displace other important and effective educational practices, and they certainly displace a significant budget that might be better spent on teachers. Stoll writes by telling stories, creating visions of creative, messy, hands-on education being displaced by sterile virtual busywork. Each of his stories is designed to invoke a headshake of empathy—the tale of the small boy whose imperfect, but unique and original paper model is ignored by adults who are agog over a child-created display of clipart. The overall result is compelling.

The reader is left seriously wondering whether all these invasions of computing make sense. Is computing, especially the simple use of computer software, really more important than art in developing a growing intellect? Is word processing honestly an academic subject worth serious school time? Does Powerpoint really make a boring talk less dull? Does it even make an interesting talk better? Stoll strives for an apparently simple, common-sense view, so that his readers will consider the broad issue, even if they disagree with his particulars.

Archive Index Issue Table of Contents

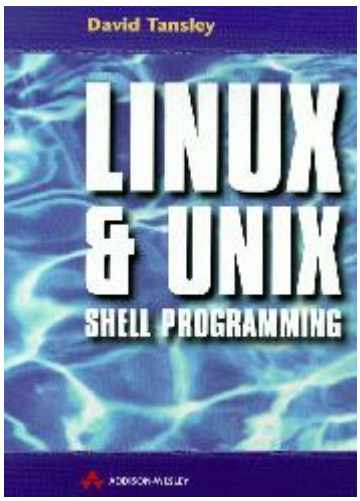
Advanced search

LINUX & UNIX Shell Programming

Marjorie Richardson

Issue #74, June 2000

This book needed better editing; the mistakes I found should have been caught.



- Author: David Tansley
- Publisher: Addison-Wesley
- E-mail: info@awl.com
- URL: http://www.awl.com/cseng/
- Price: $44.95 US
- ISBN: 0-201-67472-6
- Reviewer: Marjorie Richardson

When beginning to read *LINUX & UNIX Shell Programming*, I was a bit confused to find I was reading about shell basics and not programming. My conclusion was, Mr. Tansley is writing for the true beginner to UNIX and so is first providing background and command information that will be needed when the user actually begins writing scripts. He is covering the basic or Bourne shell, which is common to all UNIX flavors including Linux, so scripts will be portable. More than once, he emphasizes the fact that his scripts are not the most efficient design, but are easy to understand and reuse. The back of the book states all

scripts are available via an FTP site, but then doesn't give an address, and I couldn't find one in the book—strange.

The book is divided into five parts. Not until Part 4 does Mr. Tansley actually begin to talk about programming. He expects the reader to know only how to log in and use a text editor. While this was a bit disappointing for me, it won't be for someone who has not worked with UNIX for as long as I have.

This book needed better editing; the mistakes I found should have been caught. Mr. Tansley has been a UNIX system administrator for some time and clearly knows his subject. But run-on sentences with improper punctuation, extra words, omitted words and misspellings (e.g., dirrectory) make understanding the material harder than it should be. An entire chapter is devoted to the **find** command, and it contains two serious editing problems. One, when explaining the use of the **mtime** option on page 25, this statement appears:

> Use "-" to specify files that have not been accessed in *x* number of days. Use "+" for files that have been accessed.

This has the meanings of - and + reversed, as is demonstrated in his printed examples. Two, in his explanation of the **exec** option, there is this statement:

> To use exec you do have to have the "-print" option on.

Then **-print** is not in any of the examples. Obviously, the word "not" got left out of that sentence. Page 52 has a similar error: it says "If its Linux... Remember to use -n echo option", then doesn't use **-n** in the following example. A good editor would have caught these errors.

In general, the explanations are good, though not heavily detailed. The author's style is casual and simple, making it easy to read and understand. He discusses many useful things such as find, cron, grep, wild cards, pipes, input/output, awk, sed, environment variables and many shell commands. All this is a prelude to getting down to the serious business of scripting.

Part 4 covers shell scripting, and Part 5 covers better shell scripting. Part 4 begins with explanations of how to use **test** and **expr** to determine file status and string evaluation—a very good place to start, indeed. Then he moves on to discuss flow control structures (e.g., if-then-else, for, while and case), exit status, creation of menus and printing. Examples are simple and to the point, demonstrating the current topic quite aptly. A good example is how to make your script interactive; that is, accept input from the keyboard rather than run in the background. Also in Part 4, he covers passing parameters, creating screen input, debugging and built-in commands.

Part 5 goes on to advanced scripting, with more details on “here” files and how-tos for run-level and cgi scripts. He includes a goodly sample of his own administration scripts which are quite helpful. There is much useful information in these two sections to help the beginning scripter. I did not find technical errors in these parts as I did in the beginning.

LINUX & UNIX Shell Programming is a good introduction to the shell as well as scripting. With a good editor, it will be even better in a second edition. Right now, I hesitate to recommend it to complete beginners, even though they are its target audience; however, for someone who has used UNIX for a short while and has the ability to sort out any confusions caused by typos, it could provide a useful reference.

Marjorie Richardson is Editor in Chief of *Linux Journal* and can be reached via e-mail at info@linuxjournal.com.com.

Archive Index Issue Table of Contents

Advanced search

Linux Apprentice: Linux Tools for the Web

Ralph Krause

Issue #74, June 2000

A look at some tools to help you easily create and maintain your web site.

I run a small consulting business in Michigan, and one of the services I provide to my clients is web site creation. I had been using my Windows machine for this work, but was not satisfied with the tools I was using. They lacked flexibility, and I had occasional lockups and other problems. Since I used Linux for all my day-to-day tasks (browsing, e-mail, etc.), I searched for Linux tools to do my web design work.

I wanted tools for editing HTML, updating client web sites easily and keeping track of revisions. I tried to select programs that had GPL or BSD licenses or an equivalent. Also, I wanted tools with flexibility, preferring to use several instead of one monster tool that tried to do everything. Finally, I wanted tools that would compile and run on my SuSE 6.1 system without requiring me to run GNOME or KDE.

I found a number of tools, and have been using them for a month now on several client sites: some I imported from my Windows environment and some I created under Linux. The tools available under Linux turned out very capable and easy to use; I was quickly able to be productive using them. Right now, the only thing I cannot do under Linux is use my scanner, but I expect that to change shortly. [All you need is a scanner with Linux support and the program **xvscan**. —Editor]

HTML Editors

I prefer to use non-WYSIWYG (what you see is what you get) editors for the bulk of my HTML work, so I have been using Bluefish and August. While both of these editors are not up to version 1.0 yet (Bluefish is at 0.3.5 and August is at 0.52 beta), I have found that they run reliably and work well.

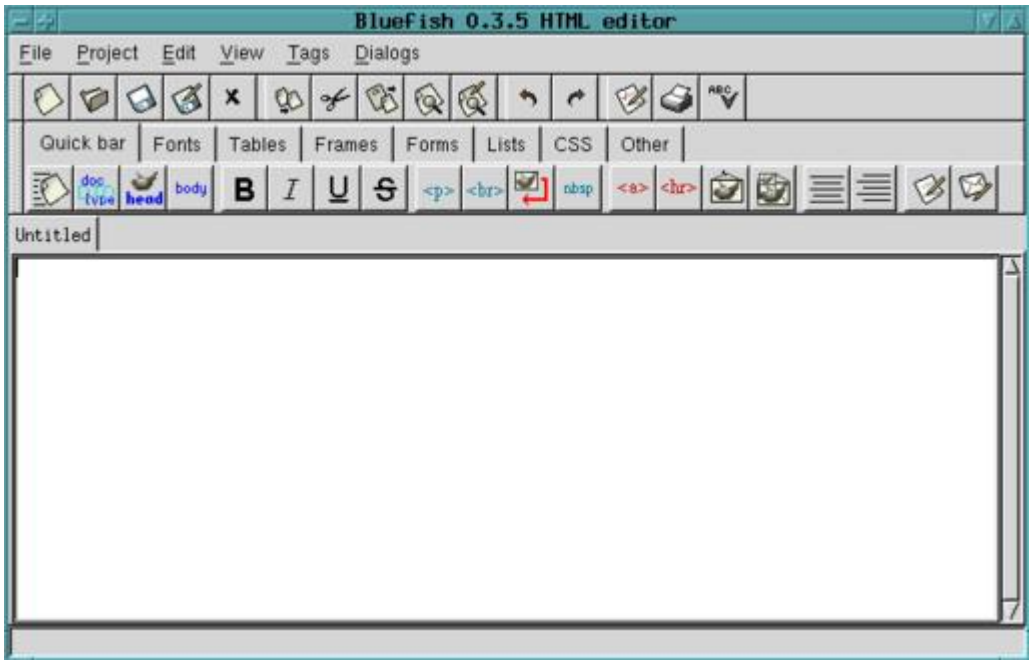


Figure 1. Bluefish Editor

Both of these editors allow you to work on multiple files at the same time. They both provide buttons to insert basic HTML tags such as headings, lists and text attributes. Bluefish uses a tabbed menu, which allows more tag selections via pushbuttons (see Figure 1) while August has a single push-button menu and uses drop-down menus for selecting other functions (see Figure 2). This makes for a cleaner interface and quicker selection of basic HTML tags. Both of these editors let you preview web pages using Netscape, and August also allows you to preview your page using Lynx.

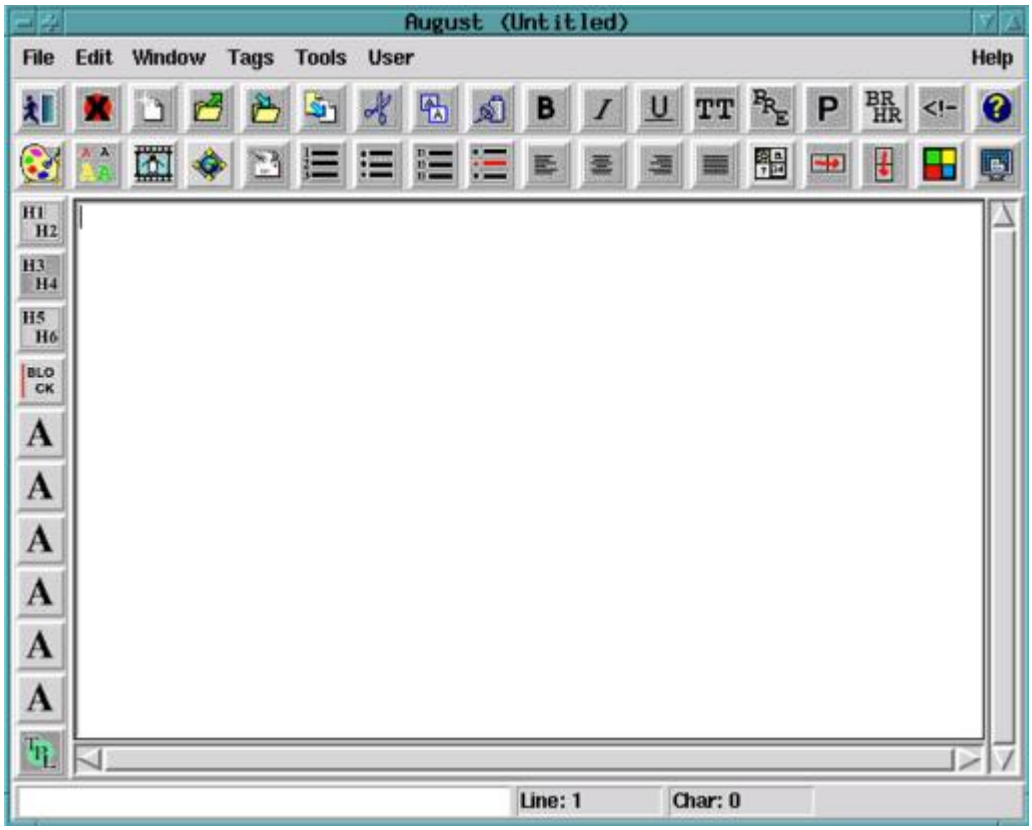


Figure 2. August Editor

Bluefish allows you to group your pages in project files. The next time you want to edit them, all you have to do is open the project from the Project menu. Bluefish also offers basic wizards for such things as creating new pages and tables.

August does not provide a project menu, but you can open all the HTML pages in a directory using a command-line switch when you start it. August also allows you to create templates and your own tag combinations accessible via the user menu. I found the user-defined tag feature to have a few problems.

Bluefish requires the GTK libraries to compile. You also need the imlib libraries for the “Insert Picture” functions to work. August is a Tcl/Tk program and therefore does not have to be compiled.

Despite being pre-1.0 versions, both of these editors have worked well for me. I prefer Bluefish so far, but not for any specific reason.

Remote Site Maintenance

Keeping remote web sites up to date manually can be a chore. Fortunately, there are tools that help automate this process. The two tools I have been comparing are **weex** and **sitecopy**. weex is currently at version 2.3.0, and sitecopy is at version 0.9.5.

To use them, first create a .rc file that contains site information such as the remote server (ftp.*here*.com), your username, password, source (local) directory and the remote directory to update. When the programs are run, they compare the contents of the local directory to the remote directory, then try to make the remote directory match the local directory.

weex offers very precise control over the files and directories that can be updated or deleted. For example, you can tell it to ignore specific directories on the local file system and other directories on the remote file system. You can provide it with a list of file names to ignore or use wild cards. When a file is ignored, it is not copied or deleted.

While **sitecopy**'s file control is not as precise, it offers some features that weex does not. For example, sitecopy can determine whether a file needs updating by checking its timestamp or calculating its MD5 checksum. This is handy if a file's timestamp changes, because you checked it out of a revision control system.

sitecopy can also notify you if someone has changed files on the remote site after you have uploaded them. When this “Safe Mode” is active, sitecopy stores the modification time from the server when you upload a file. The next time sitecopy is run, it compares that stored time to the current modification time on the server. If the times do not match, sitecopy displays a warning message and will not overwrite the remote copy, allowing you to see what changes have been made. I have not tried this function yet.

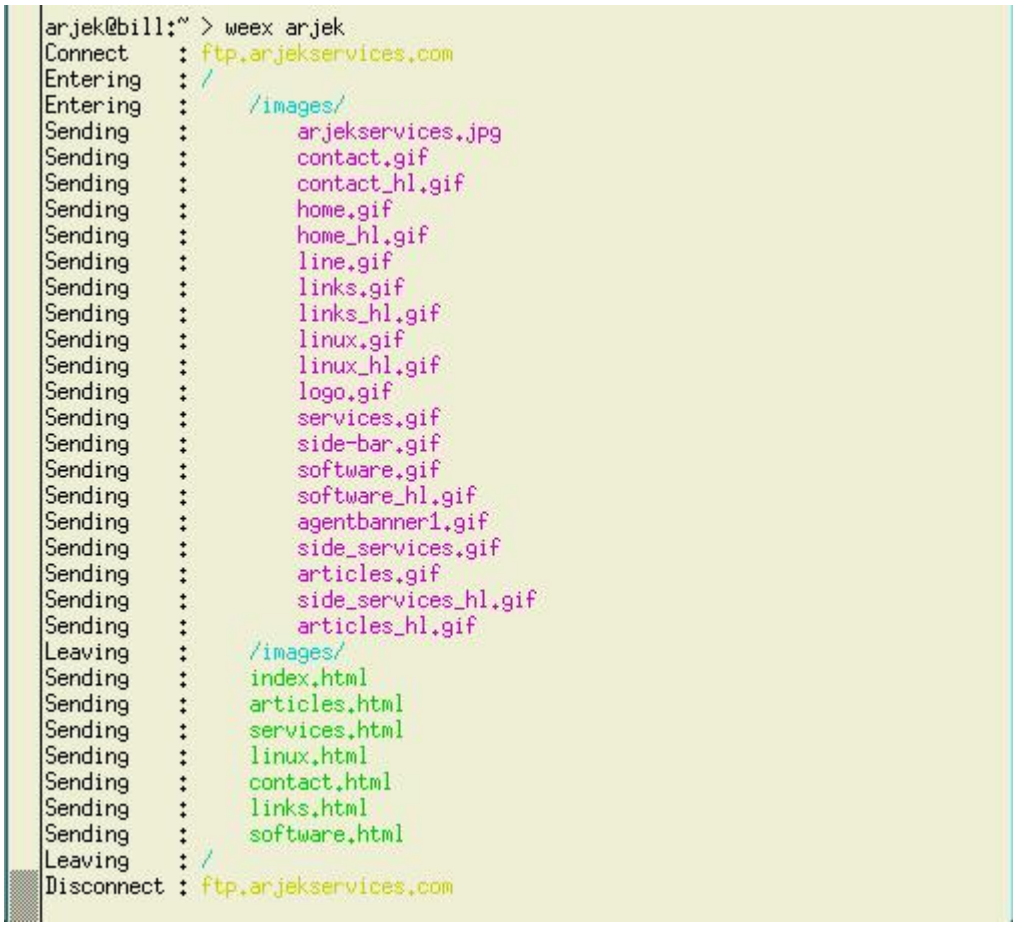


Figure 3. weex Console Output

The console output of weex is more colorful and provides a bit more detail than that of sitecopy (see Figures 3 and 4). The weex command interface is also simpler: all you do is issue the weex command followed by the name of the site you wish to update. To run sitecopy, you issue the sitecopy command, followed by the action you wish to take (update, fetch, synchronize), followed by the site you wish to use. If you do not include an action command, sitecopy displays the files that are different between the local and remote directories.

Figure 4. sitecopy Console Output

I have found the best way to use these tools is to create the directory structure on the remote machine before running them. It is also best to make a complete backup of your remote directory before running either of these programs, so you do not accidentally erase important administration or cgi-bin files.

sitecopy provides a **nodelete** option that prevents it from deleting files on the remote file system, and weex offers a **test** option that does something similar. I recommend you use these options the first few times you run the programs.

While I prefer the simpler command line and more colorful console output of weex, it does not always keep track of the files it uploads to subdirectories on the remote site. Consequently, it uploads some files again and again, even though they have not changed. It also seems to have problems when the remote system is running Windows NT, but after a bit of tweaking, I have gotten it to work.

sitecopy runs into a problem if it tries to create a directory that already exists on the remote machine. If the remote directory exists, sitecopy quits with an error message. Running sitecopy in **fetch** mode before **update** mode takes care of this.

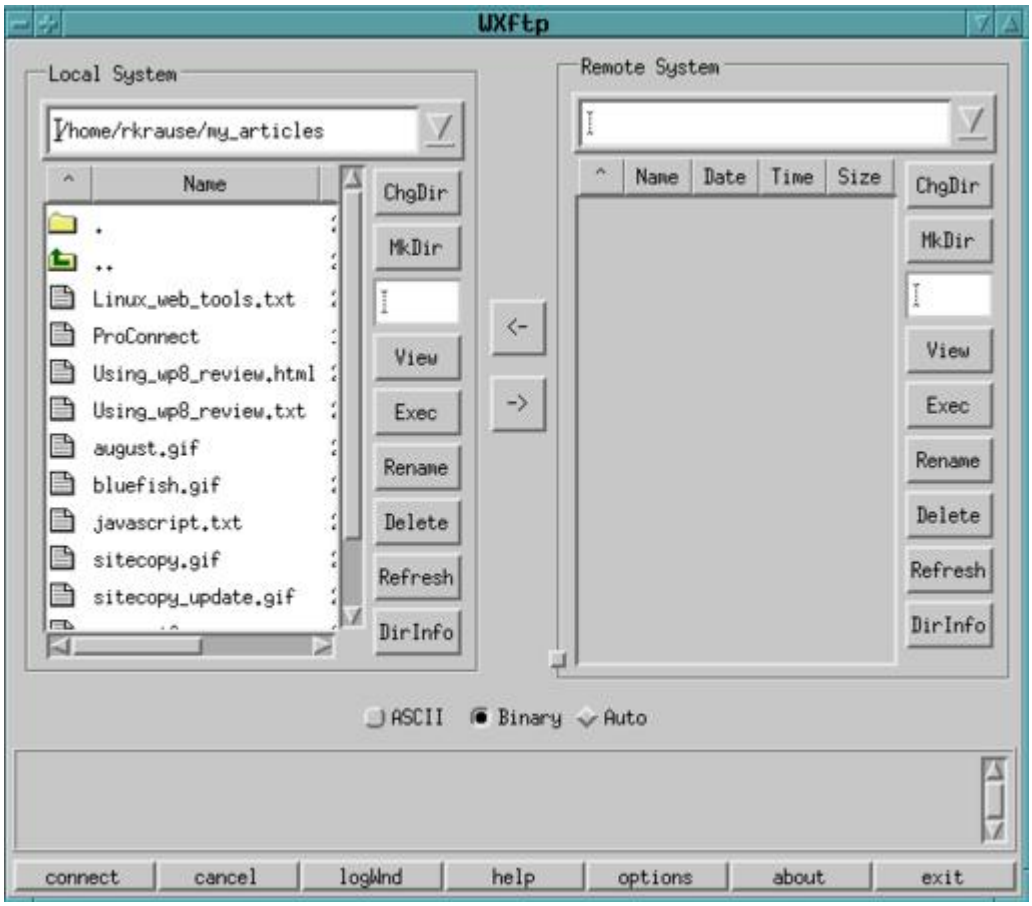


Figure 5. WXftp Main Window

For copying files manually or creating the initial remote directory structure, I use **WXftp**. This program provides a graphical front end to the **ftp** command (see Figure 5). It also stores user information for sites, so you do not have to enter addresses, usernames and passwords manually (see Figure 6). You can also configure it to switch to specific local and remote directories for each site you maintain. In addition to copying files, you can create and remove directories, delete files and execute commands on the remote system. I have been using version 0.4.4 and had no problems with it.

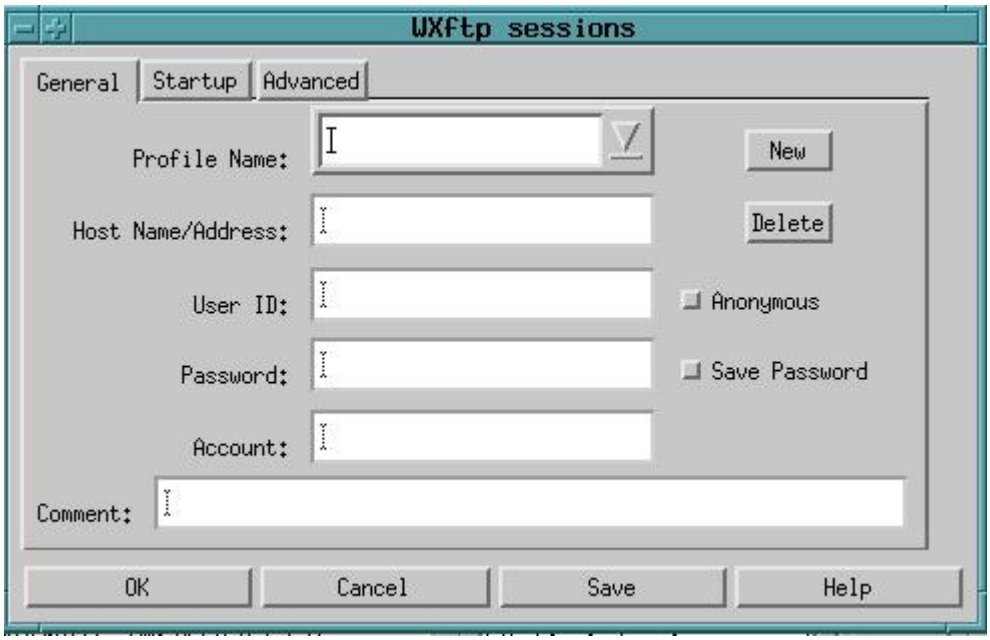


Figure 6. WXftp Sessions Window

Decreasing Drudgery

Creating and maintaining HTML code can be a difficult and boring task. Updating things such as menus and e-mail addresses by hand on several pages for multiple sites can lead to mistakes. I found a program called GTML that helps ease this task. GTML is a pre-processor for HTML files. To use it, create your page with a .gtml extension that contains GTML commands along with HTML. When you are done, you run the .gtml file through the GTML program and it creates a file with an .html extension for you.

GTML allows you to do simple things such as include text files in your HTML files, and complex things such as conditional processing. It even supports embedded Perl code or system commands.

I frequently use GTML's **#include** directive to keep from typing contact or title blocks repeatedly on my pages. Also, if I have to change something like a contact e-mail address for a site, all I need to do is make the change in one text file and then re-run GTML over the site's .gtml pages instead of making the change in each .html file.

I have used the GTML conditional operators (if, elseif, else) to build side bar menus automatically. Each page is uniquely identified using the **#define** command (e.g., **#define THIS_PAGE=home**). I then create a text file that contains the GTML and HTML code for the menu. This code checks the page identifier, then determines which buttons are active and which button corresponds to the current page. The active buttons are generated as links to other pages, while the current page's button either does nothing or is highlighted. Then, I include this file in all the site's pages and run them through

the GTML program. Now adding or removing menu buttons for a site is a simple matter of changing one text file and re-running GTML.

You can also define values to GTML in the command line. For example, you could call GTML with the following command:

```
gtml-DMY_EMAIL=me@
```

and all occurrences of **MY_EMAIL** will be replaced with **me@*somewhere*.com** in the resulting .html files. This makes it easy to generate things such as contact or copyright information for different sites using a single template.

According to its documentation, GTML can automatically generate Next, Previous, Up and Down links between related pages, as well as a table of contents. I have not used this feature yet.

I have been using GTML version 3.5.3 and have had no problems with it. Its command syntax is simple and straightforward. One thing to note is that GTML commands must be flush with the left margin in your files; otherwise, the pre-processor will not execute them and they will show up in your HTML files.

Clean HTML Code

Once I am done creating my web pages, I run them through the **weblint** program, which checks the syntax of HTML documents and flags errors in much the same way **lint** works on C programs. **weblint** can check local files or files stored on the Web using Lynx. By default, weblint checks HTML code against the HTML 3.2 standard. The program also has flags to tell it to check HTML against Microsoft- and Netscape-specific extensions.

I have been using weblint version 1.020 without any problems. From what I've seen on the weblint web site, it appears that development of weblint may be halted at this time.

Graphics

For doing graphics work, I chose the GIMP. Since I am not a graphic artist, I frequently use the Script-Fu extensions to create required graphics, then tweak them as necessary. Script-Fu extensions make short work of creating page headers and sidebar menus.

One feature I wish the GIMP offered is the ability to see how different factors (palette size, interlacing and JPEG quality) affect the resulting file size before saving the file. Perhaps there is a way to do something similar in the GIMP; if so, I have not found it yet.

I have been using version 1.0.4 of the GIMP and am completely satisfied with it.

Miscellaneous Tools

There are a few other tools I use which I have not mentioned. For revision control, I use CVS. I am finally somewhat comfortable using it, although I found it a bit difficult to understand at first. While the accompanying manual explains the CVS commands well, I think it could use more examples. I have been using version 1.10 of CVS.

For working on the text files used in conjunction with GTML or for simple HTML editing, I use the **vim** editor. The recent versions provide nice syntax highlighting and make it easy to do quick editing. I am currently using vim/gvim version 5.6.

Another program I use occasionally is called **linefeed**, a little GTK utility that converts DOS/Windows text files to Linux/UNIX ones. The version I have been using is 0.1.0.

Conclusion

A month ago, I started looking for Linux tools that would allow me to create and manage web sites. My goals were to find open-source tools I could compile and use without too much difficulty. The tools I found and used have met my needs quite well. Even though several of them are not up to version 1.0, I find the Linux tools work better than the tools I had been using in the Windows world. I was also able to sample a much greater variety of Linux tools than I would have been able to in the Windows world.

> Ralph Krause (rkrause@netperson.net) made the leap from a corporate salary to independent computer consultant this past summer. He divides his time between his business, his girlfriend Ann Marie, and his two dogs, Purdy and Dakota.

Resources

email: rkrause@netperson.net

Archive Index Issue Table of Contents

Advanced search

XVScan

Marjorie Richardson

Issue #74, June 2000

Scanning photos has never been easier.

XVScan is simply **xv**, with the added option of scanning in artwork. As a result, everything we discussed about xv last month still holds true this month for **xvscan**, including the fact that it is commercial software. Licensing is $50 US per-installed computer. So, a work group can install XVScan on the computer with the scanner, and though everyone uses it, only one license is required (site licenses available). To use, type **xvscan** on the command line. The familiar window with the red fish swimming around appears, and clicking on the right mouse button brings up the control window. Clicking on “About XV” will give you the licensing information. The difference is seen when you click on “windows”: there is now a scanner option. Click on “scanner”, and the window shown in Figure 1 comes up.

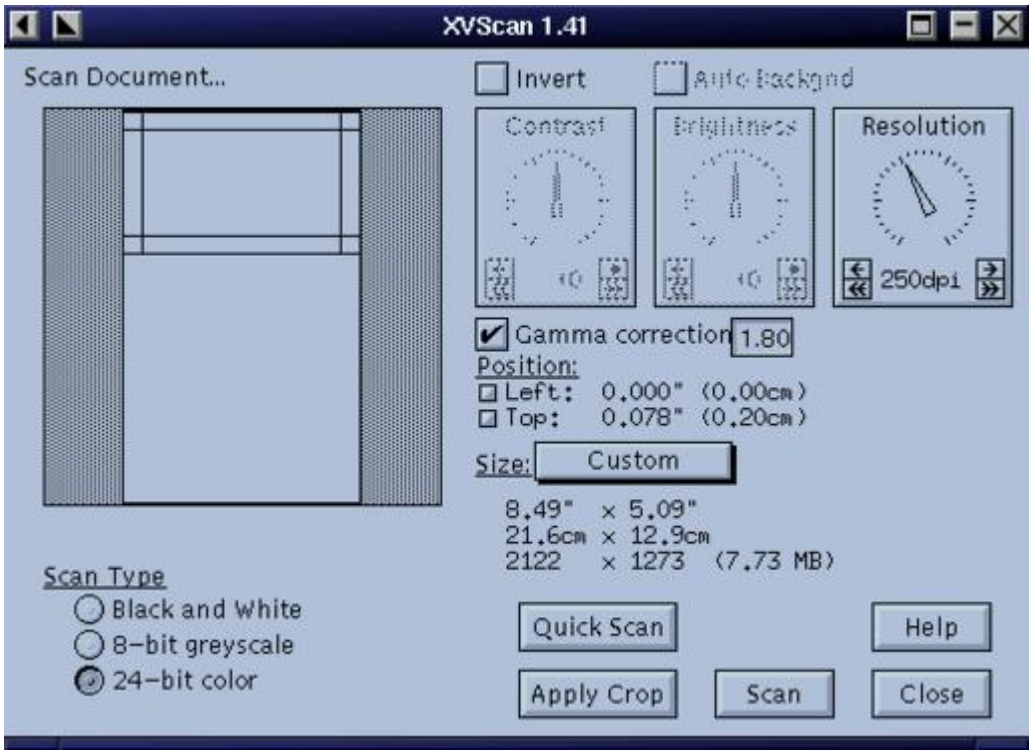


Figure 1. Scanner Control Window

The blank block under the words “Scan Document…” represents the glass portion of your flatbed scanner. The outlined portion is the area to be scanned; you can make this smaller or larger, up to the full size of the glass. I will be scanning a photograph placed at the top right-hand corner of the glass, so I will start out by scanning just the upper portion. This will make the scanning operation faster than it would be if I scanned the entire bed. It's always a good idea to check the gamma-correction button; otherwise, the picture will come out very dark. For the first scan, I just want to determine the exact position of the picture so I can crop it, so ordinarily, I would set the resolution fairly low (the lower the resolution, the faster the scan) and click on “Quick Scan”. The result of my quick scan is shown in Figure 2; I kept the resolution high, so that the picture would be printable.

Figure 2. Quick Scan

As you can see, part of the scanner bed was scanned as well as the picture. The area to be cropped is set in the same way as in xv: click the left mouse button in the upper left corner, hold it down while dragging to the bottom right corner, then release. Now click on “Apply Crop”, and the lines delineating the scanning area will decrease to the defined size. In this case, even with the gamma correction set, the picture is still a bit dark, so I turned off the gamma correction and set the contrast and brightness in order to lighten it up. Since the picture will change each time you change the values, I just played with different values until the picture looked right. The numbers I ended up using were +17 for contrast and +21 for brightness. I also clicked on the “sharpen” option under algorithms and added a label, showing that this picture of two of my granddaughters, Sarah and Rebecca, was taken in 1995. The result of all this fiddling is shown in Figure 3.

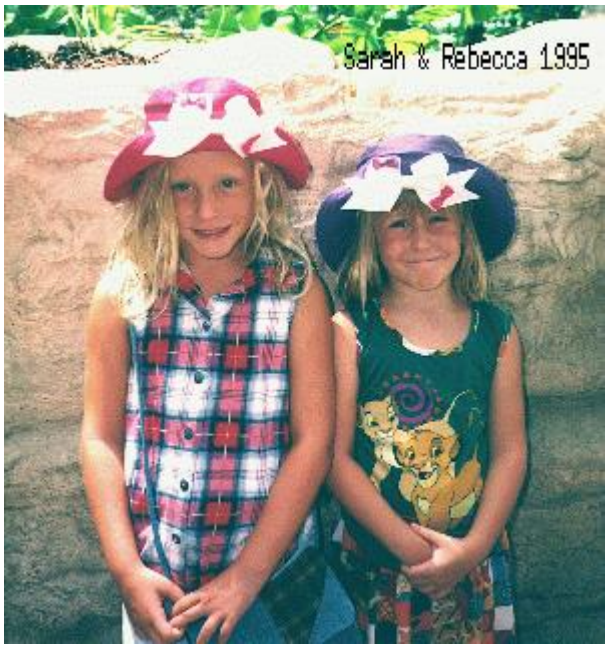


Figure 3. Final Scan with Labeling

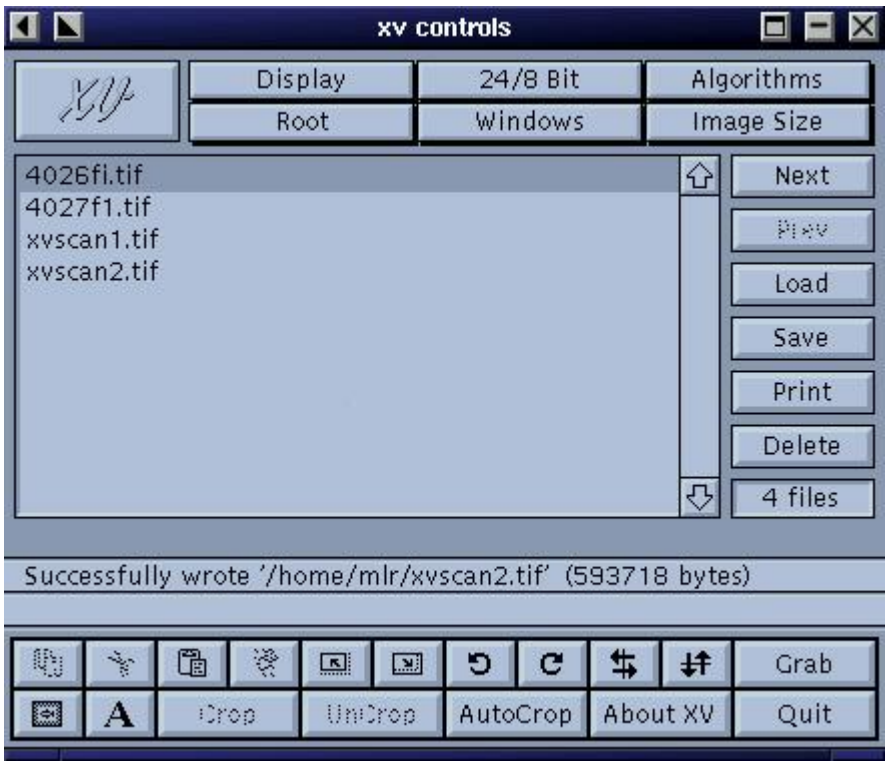


Figure 4. Control Window

For a bit of fun, I clicked on the button in the far left corner of the control window, the one that looks like a small picture of a fish (see Figure 4). This brought up the xv pad window shown in Figure 5. I had no idea what this would do, so for fun, I set it to 50% blue, and the picture turned blue (see Figure 6)--surprise! Well, that certainly showed me what it would do; more experienced photographers and artists would probably know how to use this option to further enhance their pictures.

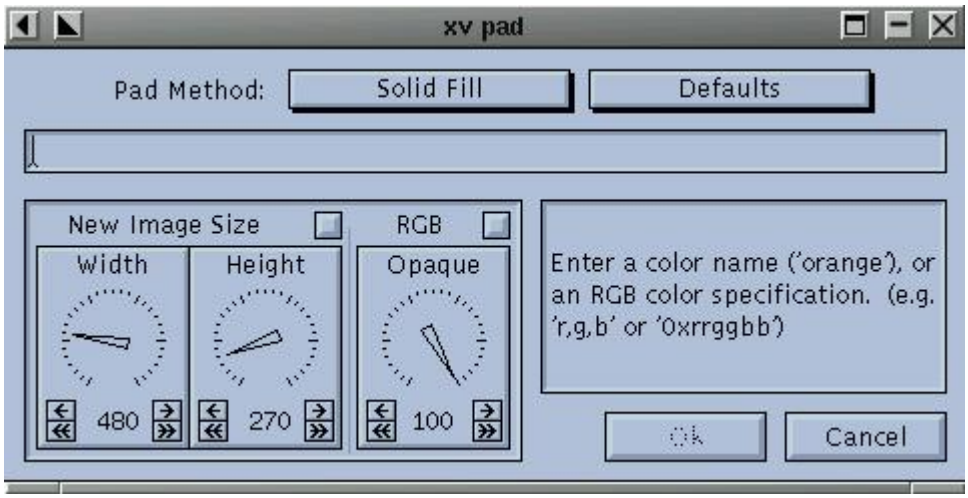


Figure 5. Color Pad Window

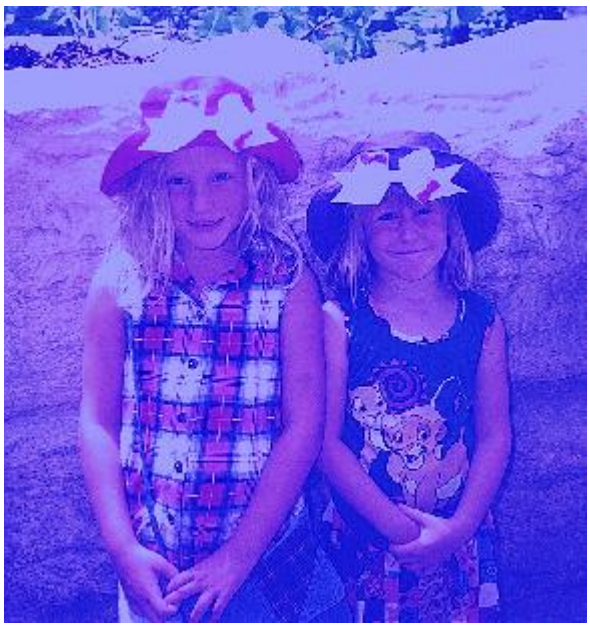


Figure 6. Photo with Blue Padding

So much for fun—that's about all you need to know to run xvscan, once you've learned the basics using xv. The problem is always what resolution to use for scanning. This, of course, depends on how you intend to use the scan. To use it only on your web site, there's no need to scan at any resolution higher than 72dpi, as that's as much as can be displayed on your screen. Scanning at a higher resolution will result in a great big file that will take a great deal of time to download, slowing down your site, *and* it won't look any better.

If you are scanning for printing purposes, then you want to use a much higher resolution. For photos we intend to print in the magazine, we need 300dpi. So, if an author sends a picture that is 1x1.5 inches, we scan it at 300dpi, as 1x1.5 is the right size for an author photo. On the other hand, book covers are printed at about 2x3, so a 6x9-inch book needs to be scanned at only 100dpi to get the right printing resolution; i.e., 600x900 divided by 300 gives 2x3. Again, we *could*

just scan everything at 300dpi, but would then end up with huge files taking up disk space, and they wouldn't print any better.

When you save, it is a good idea to check the “normal” box; otherwise, the find graphic is smaller than you expect.

Now, you too can get rid of all those bulky photo albums. Just get a Linux-supported scanner and XVScan, and scan away. Not all Linux-supported scanners are supported by XVScan, only those listed on thier web site at www.tummy.com/xvscan/scanners.html. This includes all HP SCSI ScanJet scanners from the IIc to the most recent ones, and two Microtek scanners (E3 and E6 no longer available). Many more scanners are supported by SANE (http://www.mostang.com/SANE/) a free scanner driver that works with the GIMP.

Of course, when you run out of disk space, you'll need to get a CD burner and write them to CD, but that's another Linux story. (See “Creating CDs” by Alex Withers, *LJ*, October 1999.)

Marjorie Richardson is Editor in Chief of *Linux Journal*. She enjoys both her work and her personal time. Having a wonderful family helps on the personal side. Both Sarah and Rebecca are bright and beautiful. Sarah has a black belt in Tae Kwon Do, and Rebecca is a ballerina.

Archive Index Issue Table of Contents

Advanced search

Mission-Critical Application on Linux

Rolf Krogstad

Issue #74, June 2000

This company converted to a Linux server for its Oracle database and increased operation speeds eightfold.

At Pace Analytical Services, Inc., the product we produce is data. Pace Analytical is in the environmental testing industry, meaning that our clients bring us air, water and soil samples and ask us to test them for pollutants. What we give back to the client is a report, both electronic and hard copy, that tells them the type and concentrations of compounds discovered in the samples.

The mission-critical application for our company is our Laboratory Information Management System (LIMS). This system's primary goal is to provide fast, accurate and timely information to our clients; it is based on an Oracle relational database. The LIMS handles everything, from sample check-in to invoicing, by modeling the laboratory operations. LIMS eliminates redundant processes and data entry, and allows for greater standardization in areas such as quality-control batches, data reporting and billing throughout the system.

One aspect of our Y2K program was updating the LIMS to be Y2K-compliant. This meant upgrading from 7.0.16 of Oracle RDBMS to 7.3.4, converting all the Oracle Forms 3.0 code to version 4.5 and converting C code to handle dates correctly. The real work in this project turned out to be not the Y2K issues, but rather the conversion of all the Oracle Forms code to work in the new version of the Oracle Forms tool.

Another issue we were facing at the same time as the Y2K development was response time and throughput on our LIMS servers. Over time, additional load on the systems occurred because of a larger number of users and because multiple instances of the database were running on the same server. At the time, Pace Analytical had three servers, and all were HP-UX boxes: one HP9000-H70, one HP9000-I70 and one HP9000-K200, and all had been in use for well

over three years. We were running old hardware, and our demand was exceeding its capacity to deliver.

We looked at our options for getting improved response from the hardware. We had already addressed everything we could, given the existing configuration, such as disk striping. One major bottleneck was that the entire SCSI channel was limited to a total 20MB/sec throughput. One option was to add a faster channel. HP had a card that would talk to a SCSI sub-system (RAID), but prices seemed quite high, even on the used market; the best quote we could find was about $10,000 US. Another option was to add more cards to get more 20MB/sec channels. After a close analysis, it became clear that, in addition to the SCSI channel, the hardware was also limited by the CPU and RAM.

Another option discussed was to purchase new HP-UX servers. This represented a sizable investment—well over six figures for three servers. About this time, the Pace Analytical database administrator, Michael Lester, came up with an alternative proposal. He proposed purchasing Intel-based hardware (PC-compatible) and running Oracle 8 on Linux, using Oracle SQL*Net to communicate between the current HP-UX server and the Linux database server. What Michael proposed made sense for a number of reasons. First, if we could move to an Intel-based hardware platform, it would be easier and cheaper to upgrade hardware as demands exceeded the hardware capacity. Intel-based hardware is much cheaper, and is readily available locally in case problems are encountered with components. As hardware is upgraded, an old server can be put to service as a Windows-based desktop system, but the old HP hardware is of no use to us.

The new hardware uses SCSI Ultra2, also a wide format, and each channel has an 80MB/sec throughput. Because of the low cost, we could afford to put in four channels: one for the OS and archive drives, one for the slower tape and CD drives and two for the database. The six drives for the database are configured for RAID and are split—three drives on each of the two database channels. In this configuration, the maximum throughput for the database drives would be around 160MB/sec, an eightfold speed increase over the old hardware. The CPUs are 600MHz dual Pentium III processors. In benchmarking, this turns out to be only about two times faster than the HP RISC processor on the old hardware.

The total cost of one server, including 512MB RAM, was about $10,000. We were able to build a brand-new server with eight times the throughput on the SCSI channel and twice the processor speed for the same cost as upgrading the SCSI subsystem on the existing server. We had conservatively estimated a five to tenfold improvement in performance over the old hardware. The true test,

though, is always how a system performs in a live production environment. We were not disappointed. Several report programs that, on the old hardware, would run in the background for 45 to 80 minutes now finish in six minutes or less. One lab supervisor literally ran into the IS area and asked, “What's going on with the LIMS? It is *flying*!”

The project was not as simple as just building a Linux server. As I mentioned in the beginning, the original plan was to upgrade to Oracle version 7.3.4 on the HP hardware. In order to run Oracle on the Linux OS, we were forced to go to version 8 of the Oracle database. There were no real issues installing Oracle 8 on Linux, with the exception of a minor patch that had to be installed to get it recompiled. The main problem we found with Oracle8 was a newer “rowid” concept it used.

The software conversion to Oracle*Forms 4.5 had to happen regardless of the hardware platform, since version 3 was no longer supported by version 7.3 and later of the database. We opted to use a character-based version of the Oracle*Forms 4.5. To switch to an event-based GUI interface would have meant a wholesale restructuring of the 100+ Forms programs. At first, most of the problems were syntax errors, things that were valid in Forms Version 3 but no longer legal in the new Forms version. A few naming conventions changed, and we happened to have used names for objects (fields/triggers/variables) which where not allowed under the new Forms. Then came problems with “integrating” all of the forms back into the LIMS product. For example, many of the Forms check to see which form/menu called them, and thus operate in a different mode. The function to return the calling name returned it in lower case under Version 3, and upper case in Version 4.5, so the IF statement failed. We added the **lower(xxxxx)** function to force it to lower case. Also, there are a number of forms that get called from others as pop-ups, and many of the screen layouts were messed up.

We also encountered problems with the HP character mode terminal emulation, which caused the Oracle/HP resource file to become corrupted. Eventually, we had to switch to VT220 emulation, and ran into some issues with the terminal emulation. The VT220 terminals had a strange kind of “screen” memory. Two text screens could be displayed, and a user switched between these by sending codes to the screen. The terminal emulator added a vertical scroll bar, making it possible to scroll back to see what was previously on the screen. This kept messing up scroll regions on the screen, which is what vi and Oracle use to show you a pop-up window. The emulator vendor, Minisoft (http://www.minisoft.com/), was very responsive and corrected the incompatibilities we encountered using their VT220 product.

An additional software complication is that a number of the functions in the Forms code are actually Pro-C routines called from inside the Forms session, known as “User Exits”. These functions were written specifically to run on HP-UX. Because of the complexity of deciphering, converting and testing the User Exits, and the immovable nature of the January 1 deadline, Michael proposed maintaining the HP-UX server to act as a client-side server for the Forms programs and to have it communicate to the database using Oracle SQL*Net. Our ultimate goal is to eliminate the user exits and have them written in native SQL and to convert the character-based interface to a GUI interface. This will make it possible to run three-tiered client-server on all Linux servers, eliminating our aging HP-UX hardware.

I have already discussed one measure of success of the project, the dramatic improvement of the speed of the system. Another measure that must be considered is up time. For people who question whether Linux should be used in a mission-critical application, our project is proof that Linux is up to the task. We installed the first server in August of 1999 and the other two followed one month apart, so one server has been running for five months, one for four months and one for three months. In those twelve months of operation, we have had absolutely no down time due to the Linux operating system. Several issues occurred with the Oracle database configuration, but that is not unexpected in any new installation. We also had an issue with a network interface card on one of the servers, but we replaced it with a Linux-supported card and the issue has not resurfaced.

While rereading this article, it occurred to me that there has been very little mention of Linux. In reality, that has been the case. The true measure of an effective and robust operating system is that it should be fairly transparent to the operation, and that has certainly been true in our application. Without the option of moving to a Linux operating system, our options were either to spend five times as much on new RISC systems, or limp along with much less than optimum performance at a cost equal to what we ended up spending for state-of-the-art hardware. In the case of this project, Linux was the lynchpin around which everything else was built, and it has performed beautifully.

email: rolf.krogstad@pacelabs.com

Rolf Krogstad (rolf.krogstad@pacelabs.com) is Director of Information Services for Pace Analytical Services, Inc., in Minneapolis, MN, with over 18 years experience in the industry. He is a retired violinist, having performed professionally in symphony orchestras in Vienna, Austria and Mexico before becoming a programmer.

Archive Index Issue Table of Contents

Advanced search

Secure Logging Over a Network

Federico

Christian

Issue #74, June 2000

Log system activities over a TCP network securely by interfacing the existing syslog dæmon with secure shell using simple Perl scripts.

Undoubtedly, one of the most important tasks for keeping a networked system safe is monitoring its activity. Most system programs today communicate with **syslogd** noting important events (such as an **su** request). Also, the kernel can note hardware failures and other things on that level. Of course, when monitoring for possible break-ins or trying to track down the path an intruder used to violate a system, the logs can be a precious resource. However, it is evident that an intruder could manipulate system logs if he were able to get the necessary permissions, thus completely fooling any attempt to track down the intrusion, even to the extreme point of making it hard to spot.

A first step toward solving the problem of keeping the system logs on the system itself was made many years ago when the ability to send logging messages over the network to another host was added to syslogd. Sending the needed log parts to another host, which is quite secure (a dedicated machine, for example), was actually a half-solution to the problem. Using a trivial protocol like UDP to transmit data over networks made it very easy to spoof, making it possible for the intruder to add malicious messages or even try to create a denial-of-service attack. Again, all data is sent through the network in clear form, making it simple to sniff, so an intruder could get more details about the system before attempting a break-in.

A solution we developed to solve this problem is having the standard syslogd facility use the secure shell (**ssh**) package to forward the logs, using authentication and encryption, to another computer. This way, the system we want to keep logged won't require special changes, and the system that will maintain the logs will not require another running syslogd. It will require **sshd**

(secure shell dæmon) running with some normal system tools on a normal user account.

We will present two technically similar but conceptually different methods, with some pros and cons, of performing this important task. The first could be compared to a “push” technology; that is, the machine that wants to log its data on an outside machine will actually force data to go on the other side. The second method will use a “pull” technology, since the remote client will try to retrieve the logs from the desired machine.

Requirements

Since our solutions use the standard syslog facilities to grab messages, you need a syslog dæmon that supports logging output to named pipes. Most recent syslog dæmons have this (check the man page for support, or try it directly). It is very important that you have a syslogd that opens pipes in non-blocking mode.

Older syslogs used to open the pipes in blocking mode, so if the named pipe was full (in our context, for example, if the connection had been broken for some time), then the whole syslog process stopped while waiting for the pipe to have some space available. This, of course, could have dramatic consequences, since all the logging, not just the remote one, would have been compromised. As a practical test to see whether your syslogd is handling pipes correctly, configure the whole system, then try writing in the pipe until it reaches the maximum dimension, usually 4KB. When the connection is down, check whether syslog is still logging something or if it seems to be frozen. Anyway, we suggest getting the latest version you can find (which should correct other bugs as well) and installing it on your system (at the moment, the latest version is 1.3-31). The more advanced syslog-ng package should work as well with our solution. Apart from this, you will need a Perl interpreter, the **tail** command from the textutils package, and of course, the secure shell package.

The First Solution

The theory behind this “push” solution is quite straightforward. A Perl script will run in the background, and will connect (using ssh) to the host where we wish to keep the copy of the logs. The Perl script will then monitor a named pipe we will create, and send anything it will get from the pipe over the secure connection to the other side, where we will put everything in a file. Instructing syslog to send the desired logging events to our named pipe will make the work complete.

The Glue: the First Perl Script

Let's give a brief technical description of the source code (see Listing1). Once started, the script will open a local log (defined in **$local_log**) to report failures (this can be disabled by undefining the variable), then it will try to connect to the remote host (defined in **$host**). The connection subroutine opens the ssh command to the remote side, specifying the user name to be used, whether to use compression and to establish a quiet connection. As you can see, ssh will ask the remote end to execute a **cat** of a given file, so everything will be stored there for further use. Once connected, the script will open the local named pipe we created, then forward everything to the other side, where the cat command will store it. Of course, if the secure connection should die for some reason (technically, if the print on the file handle should fail), the script will try to establish the connection again.

As a last feature, the script will send a timestamp each defined time interval to the other end, which could be useful for tracking down when the last message actually came from the host if something serious has happened. In fact, the script can detect that the connection went down each time it sends something over the network. By sending a timestamp to the FIFO (i.e., first-in/first-out pipe), it will automatically trigger the check that the connection is alive at that moment.

Listing 1

Configuring ssh on Server and Client Side

Even if this should look like the easier step to accomplish, there are a few things you must pay attention to, so that an intruder, who has discovered the remote logging facility, won't be able to use it as a way to break in to the logger machine. First of all, you must generate your key pair on the network machine from which you're sending the logs by using the **ssh-keygen** utility. You do not have to set a password (just press return twice at the prompt) to this pair of keys (that should be used just for the logging program), so the script will not have to prompt you for it each time it starts.

With **ssh2**, you'll have to specify that you're going to use this newly created private key as a possible key to identify yourself. Add it to your ~/.ssh2/ identification file by adding a new line with the **IdKey** statement, followed by the key name. (Alternatively, you could force this on the command line to ssh in the script.) Now, you'll have to transfer the public key to the system where you're going to store the logs, and add it to that user's authorized key file. (Usually ~/.ssh/authorized_keys in ssh1, where each line contains a key. In ssh2, copy the key to your ~/.ssh2 directory, then add an entry to your authorization file, ~/.ssh2/authorization by default, in the usual syntax **Key key_*filename***.)

This instructs sshd on that machine to authenticate anyone using the private key that couples with that public key without further checks.

Depending on the ssh configuration, you should normally have to log only once using this new generated key. At the first connection, ssh will have to add the key of the new host to your known_hosts files (by default, not an automatic task); it will prompt you to confirm that you would like to add the key. Pay attention to the authorization and key files permissions: they must be readable and writable only to the user—for security reasons first of all, but also because ssh will refuse to accept connections with such badly configured keys. Try logging in verbose mode if any problem occurs, since this will tell you exactly why the connection was not permitted.

A complete login over ssh could be done this way with that key. This would be very dangerous, so finally you must put some important restrictions on the use of the key. First, specify from which hosts this key could be used with the **from=** directive, so if someone were to steal the private key, he couldn't log on from any place other than the ones you specify. Second, inhibit the allocation of a **pty** with **no-pty**. Finally, limit the use of this key to just appending something to the logging file; that is, the command the Perl scripts execute on the remote machine. This is done using the **command=** directive and should be:

```
command="cat >>
```

At this point, you can be quite sure that nothing more than appending to the log file can be done, and you can use your normal account on some machine as a log storer, since this won't open you to any danger (other than having your space filled with logs), if the log key is correctly restricted by ssh. In ssh2, you'll have to add just the command restriction option, using the Command directive in your authorization file. Check out the man page for exact information on the syntax.

Since many sshd installations check the TCP wrappers files before allowing a connection, make sure the hosts from which you wish to connect are allowed to do so. As a user, be sure you don't have any .shosts or .rhosts files from the machine you are logging. If present, everything would seem to work (since authorizations would be done through these, if allowed by the dæmon), but the system would be very insecure.

Configuring syslogd

The first thing we have to do is create the named pipe needed for the communication between the syslogd and our script. This is easy; use the **mkfifo** command with the full path name of the named pipe as the parameter to

create it. Now you must tell syslogd the kind of messages it should send to that pipe, using the standard syslogd syntax:

```
   log_facility:    | /
```

It would be a good idea to send over the network the usual authorization tasks (auth.* and authpriv.*), critical messages (*.crit and *.emerg) and important kernel network-related messages (especially if you're filtering packets or something like that). It is very important to choose all the needed facilities to have a good overview of the system status. This changes, of course, from system to system (the facilities used could also be compilation, and thus distribution-dependent), so be sure to test it thoroughly.

Make It Run

Once all the steps are done, you must start the script, redirecting the standard error and output to /dev/null so ssh errors won't be displayed. Then, once it runs, restart syslogd so it will reread its configuration. If everything has been done correctly, you should now have your logs on the remote machine, and you should put the script in your startup file so it will be started at boot time. The program can run using normal user privileges; it doesn't need any special permission. Just make sure it can read from and write to the named pipe created for the communication with syslogd, and eventually, to the file selected for local error logging.

The Second Solution

The second method is based on retrieving the data, a “pull”, instead of receiving it from the syslog. In this solution, a Perl script will run on the machine where we are going to store an additional copy of the logs, which will connect to the machine we want to monitor and obtain the logs from it. This task will be done just by using the **tail** command on the log file, so everything that comes to it will also be displayed on our side.

The Glue: the Second Perl Script

This script is quite similar to the first one, as you can see (in Listing 2). It has the usual procedure to connect to the remote host using ssh, the variables that hold the configuration and some local logging capabilities. Since the script is running on the machine that holds the additional copy of the logs, the logs for the script will also be stored on this machine. Once the connection is open, the script will continue reading from the ssh file handle and print everything on standard output, which will very likely be redirected to a file on startup. As we invoked it, ssh will execute tail on the log file, so each time something is added to the log file, it will be sent over. If an error occurs, then the script will try to reset the connection and log the error.

Listing 2

Configuring ssh on Server and Client Side

The configuration of ssh is very similar to the previous, but of course the roles are exchanged. In fact, here we will have to connect from the additional logging machine to the machine we want to log, so you will have to create the key on the additional machine and copy the public key to the authorized keys on the other machine. All other restrictions and hints should be used in the same way as before. Remember, you must now set the command to execute to the new one; that is, the tail defined in the Perl script by the **$cmd** variable.

Configuration of syslogd

In this example, you shouldn't have to change anything in the syslog, since you will be using tail on the already-existing log file. Anyway, we strongly suggest creating a new log file (using the usual syntax) including the most important things that need to be stored on the other machine. Another small problem arises from the fact that this consumes disk space. This is quite easily solved, since you can add a job to your crontab that deletes the file from time to time. Remember that syslog must be restarted if the log file has been deleted and recreated using **touch**, since it must exist for syslog to use it. If you have a recent version of textutils, then tail (with the **--retry** option) won't care about the file being deleted (use the **rm** command) and will create and open a new file. If you have an older version that doesn't support this option, you could just **kill** the tail command (this is easily done with some piping from **ps** to **grep** to kill) in the same cron as before, and the Perl script will then restart it from the remote side. Of course, we strongly suggest upgrading the textutils if possible.

Make It Run

Once configured, you should just start the script, redirecting the standard output (and error, too, but this isn't necessary) to the file that will keep your logs. The script on the logging machine doesn't need any special privileges. Of course, the file with the logs must always be readable by the user who is effectively running the tail command executed via ssh in the remote script. This means that if you're going to delete the file, you must set the correct ownership and modes each time in the cron job.

Comparison of the Two Methods

Both ways are effective for remote logging, and they are almost equivalent. We wanted to describe both, since one could be easier to use in some environments. For example, if you don't have sshd running on the machine where you want to store the logs, you could just install the ssh client in your

home and use the second solution. Also, we wished to point out some interesting problems that can arise in communicating over a network.

The first method uses a somewhat “real time” logging. Each time a message is generated by syslog, it is sent through the pipe to the other side if possible. So, once something is generated by syslog, it is sent (in normal circumstances, of course) and can't be stopped in any way. This is a great feature for such a system, but causes a general process communication problem: buffering. Since the pipe that serves as a communication link between syslogd and the program was opened in non-blocking mode and has a finite buffer size, if syslog generates data too fast, then the data will be lost. If it could open it in blocking mode, we could solve this problem, but then if the pipe couldn't get free space in a short time, the process would just freeze and lose other data. This is an interesting problem of real-time communication. Practically, it shouldn't be a problem, since if you intelligently select which syslog facilities to send over the network and a decent connection exists between the two machines, then the buffer shouldn't get filled so quickly.

As a way of solving this possible data loss, the second program will read the logs from a disk file, jumping over the buffering problem (the disk file is the growing buffer, so it doesn't have such restricted limitations). Of course, this isn't a “real-time” solution, since tail has to check whether the file changes in given time periods, otherwise it would constantly be using the CPU to check for changes; see the **--sleep** interval in the info file. So theoretically in this short time period, a malicious cracker should find your program running and kill it (again, very improbable).

Security Implications

As to the security of the account used for the secure shell, the two methods are both secure if the restrictions are set correctly as explained. There shouldn't be a direct way to break from one machine to the other by using the restricted logging access, since nothing but the designed command can be used. Of course, you should have noticed that once a break-in is made on the machine we are logging, the cracker can initiate a denial-of-service attack by trying to fill the disk space on the remote machine. This is an old problem with logging and can be dealt with simply by placing the logs on a non-vital (that is, a non-root) partition or by using quotas if possible. Since the script doesn't need any other special permission to be executed, it is also evident that there aren't any root-privilege process problems.

Federico (drzeus@infis.univ.ts.it) is studying computer science at the University of Udine. When not hacking or coding, he enjoys reading science fiction, listening to music and playing guitar.

Christian (chris@infis.univ.ts.it) is studying astrophysics at the University of Trieste and works part-time as a system administrator and high school teacher. When not playing with Linux and other fun software or hardware, he enjoys discussing who is the best film director of all time with his girlfriend.

Archive Index Issue Table of Contents

Advanced search

Contributing to the Linux Kernel

Joseph Pranevich

Issue #74, June 2000

To make changes in the kernel, you need to know all about the diff and patch commands.

Everyone who knows about Linux also knows about the ways Linux is “different” from other, more commercial operating systems. Because the Linux kernel is open source, it is possible for each and every user to become a contributor. Certainly, nearly everyone reading this knows so; it's sort of like preaching to the choir. However, the fact is that most Linux users, even those skilled in the programming arts, have never contributed to the code of the Linux kernel, even though most of us have had days where we thought to ourselves, “gee, I wish Linux could do this...” Through this article and others, I hope to convince some of you to take a look at the Linux kernel in a new, more proactive light.

What are some valid reasons for not contributing to the kernel efforts? First, maybe you can't legally. Many programmers sign contracts that limit their ability to code outside of work, even on non-commercial projects. This is the main reason I chose a profession that has relatively little to do with programming, other than the occasional Perl script. Second, it is possible you don't know how. Many Linux users are relatively new programmers trained in traditional computer science. I know from my own CS education that many schools tend to teach “modern” programming skills—I was one of the few in my particular school who chose (or knew how) to program without an IDE (integrated development environment). Sad, but true. Third, many professional programmers now tend to work with revision control systems in the workplace, and may be hesitant to contribute to projects (such as the kernel development effort) which still use the “bare metal” approach. Last and most likely, many programmers with the skills to hack Linux don't have the time to do so. These are all valid reasons why perfectly qualified programmers with good ideas, a fresh outlook and a desire to contribute have chosen not to. Nothing I can say

can help them get past some of those issues, but I hope I can make kernel programming more accessible to at least a small percentage of people.

This is the first article in a series, and I will attempt to dispel some of the mystery behind revision control. Many open-source projects, including the Linux kernel, still use the **diff** and **patch** method of content control for a variety of reasons. Most open-source projects still accept patches in this format, even if they distribute code via CVS or some other revision-control system. First, diff and patch provide a project maintainer with an immense amount of control. Patches can be submitted and distributed via e-mail and in plain text; maintainers can read and judge the patch before it ever gets near a tree. Second, there's never a worry about access control or the CVS server going down. Third, it's readily available, generally doesn't require any special tools that aren't distributed as part of every GNU system, and has been used for years. However, bare-bones revision control makes it difficult to track changes, maintain multiple branches or do any other “advanced” things provided by Perforce, CVS or other revision control systems.

diff and patch are a set of command-line programs designed to generate and integrate changes into a source tree. There are multiple “diff” formats supported by the GNU utilities. One major advantage of diff and patch over newer revision-control systems is that diff, especially the *unified* diff format, allows kernel maintainers to look at changes easily without blindly integrating them.

The diff Family

For the uninitiated, diff and patch are just two of the commands in a complete set of GNU utilities. While they are the most commonly used in practice, other tools are often employed in specific situations. For the purposes of this document, I won't concentrate on these utilities, but will treat them only briefly. For a more complete look, check out your local set of man and info pages.

diff is the first command in the set. It has a simple purpose: to create a file (often confusingly called a patch or a diff) which contains the differences between two text files or two groups of text files. These files are constructed to make it easy to merge the differences into the older file. **diff** can write in multiple formats, although the unified difference format is generally preferred. The patches this command generates are much easier to distribute than whole files, and they allow maintainers to see quickly and easily what changed and to make a judgment.

patch is diff's logical complement, although oddly, it didn't come along until well after diff was in relatively common use. **patch** takes a patch file generated by diff and applies it against a file or group of files. **patch** will notify the user if

there is a conflict, although it is often smart enough to resolve simple conflicts. Additionally, patch can act in the reverse; with an updated file and the original patch, this command can revert a file back to its pristine form.

cmp is diff's counterpart for binary files. As applications for binary files in source control are limited, this command is often not used in that environment. Usually, projects that include binary files (for example, a logo) have some other mechanism for updating these components. (Keep in mind that the XPM image format common in Linux applications can actually be text-based and can be controlled using the above commands.)

diff3 is a variant on diff that allows for computing and merging the differences among three files. Personally, I tend to use diff for these purposes, but there are likely reasons why this command is useful in specific situations with which I have not yet dealt.

And finally, **sdiff** is an interactive version of patch that allows for smarter patching using your very own brain.

These tools have many uses other than content control. I do not want to slight them by implying they are not useful. But, like many tools, they shine only in certain circumstances. (Like that annoying fine screwdriver you get with the set for which you've never seen a screw small enough to use it on, tools are only as good as the situations they are applied against.)

diff Concepts

When talking about different patch formats, a number of concepts deserve some thought. The point is probably moot: nearly all open-source projects that use diff and friends have long since settled on a patch format. However, these are some of the qualities in a patch that makes one format more useful than another.

The first thing admired in a patch format is context. Context consists of the extra lines (often three) before and after the difference blocks in a patch. While context adds greatly to the size of the patch file, it allows patches to be not entirely dependent on the exact file on which the patch was based. This quality is very important in a revision-control system, where it is expected that your working files will be slightly different from the master copy. Patching programs can use these context lines to guess where the offending lines can go—and usually get it right.

Second, patches should be reversible. Reversing is useful when you want to go back to a previous version of the source, or when you mistakenly flipped the

options to diff and generated an inherently reversed patch (don't laugh—we've all done it). Not all patch formats are reversible, however.

Patch files should be efficient: small and easily readable, but not so large as to be unwieldy for projects with large numbers of changes (such as the Linux kernel). There is a tradeoff here, of course. Patches without context are more efficient, but definitely not very useful in source maintenance.

Finally, readability is a very important aspect for this style of revision control. Patches should be obvious when it comes to figuring out what changed and should not require the user to flip his or her perspective from one block of text to another to figure out the differences. Making a format human-readable is much more difficult than creating a computer-readable one, especially when you are trying to balance all the other format components.

Each of the various diff formats ranks differently in each of these metrics, and different projects may choose different formats. The Linux kernel, for example, uses the unified difference format.

The diff Formats

The POSIX, or “normal”, diff is the default format used by the diff utilities. It's a terse format without any lines of context, but is reversible. I've often seen it used as a sort of “generic” patch format, since non-GNU versions of the diff utilities can parse it.

The context diff format is another reversible format similar to the POSIX one, but which supports context around changes. Some projects prefer this format, especially if they include developers outside the GNU sphere. Using GNU diff, this format can be specified with the **--context** option.

The unified diff format is another contextual format that is generally more readable than the context variety. Unlike with context diff, this format displays all the changes in a single block, thus eliminating many of the redundant lines with context diffs. Because of the relative merits of this format for revision control and easy reading of patches, this is the preferred format for the Linux kernel and many GNU projects. On the down side, many non-GNU patch programs are unable to recognize this format. With GNU diff, this format can be specified using the **--unified** option.

The side-by-side format is great for human reading of patches, but is not readily usable for revision control. It displays the originating file and the changes side by side. This format is mostly just for human patch browsing, and the patch program doesn't actually support it. GNU diff users can enable this with the **--side-by-side** option.

The ed format is an old format that outputs a script for the **ed** text editor rather than a special patch format. This option was needed before the modern patch utility was created. Since it outputs a script, it contains no context information and no reversal information. GNU provides this option only for compatibility, and it may be invoked with the **--ed** switch.

The forward ed format is similar to the ed format, but is even more useless. **patch** can't process files in this format; neither can ed. If you insist, however, GNU diff still can generate it with the **--forward-ed** option.

The RCS format is the one used by the revision-control system RCS and its derivatives. It's generally not used standalone, and patch can't actually read the format.

The preprocessor format is not quite a patch format and not quite a script. Instead, it is an output file that contains the contents of both files separated by C preprocessor directives such as #ifdef, #endif, #elsif, etc. It is possible to compile either version of the file by setting preprocessor variables (#define) in the source file or by using the **-D** switch of the GNU compiler. Obviously, this format isn't of much use for revision control, but can occasionally be useful when you want to test changes.

Using diff

The actual usage of diff and patch is not complicated. While these commands, like many GNU tools, support many options allowing users to refine the way these tools work, these options are not actually required for everyday use. For more information on all the command-line options of these utilities, check out their info pages.

At its simplest, a diff command line for comparing two files would be:

```
diff old.txt new.txt > oldnew.patch
```

This will create a patch in the POSIX format that could later be applied to files similar to old.txt. Please note that the output of diff generally appears on standard output (STDOUT) and we have used redirection to get that information into the patch file. For GNU projects, we generally want the results in unified diff format, so we add the **-u** (or **--unified**) option to the command line:

```
diff -u old.txt new.txt > oldnew.patch
```

Generally, however, a comparison of two source trees is often desired. These trees would be multiple revisions of a single project or something similar. The command I generally prefer for this would be:

```
diff -ruN old new > oldnew.txt
```

In this example, I have added two new switches. The first, **-r** (or **--recursive**), indicates we want to take a recursive look at directories instead of files. The last switch, **-N** (or **--new-file**), indicates we do not want to ignore whether files have been added or removed from either set. In that case, if the new directory included a file called foo.txt but the old directory didn't, the patch would behave as if there was a zero-byte file called foo.txt in the old directory and add it into the patch.

To actually get good use out of the diff command as a form of revision control, a bit of legwork must be done first. I'll discuss this later on.

Using patch

Generally, once a diff is generated or downloaded, the process of patching the file is even simpler. Based on our first example above, we could do something like this:

```
patch < oldnew.patch
```

This command would read the patch file from the standard input and apply it to whatever files were in the current directory. Most patch formats include information on the name of the file being patched. In our first example, it would have specified that old.txt was the original file, and the patch command will look for a file by that name here. If that file could not be found, it would then prompt you for the name of the file to which the patch should be applied.

A number of things can go wrong during the patch process. Occasionally, a diff may be made backwards, or you may want to reverse a patch. By using the **--reverse** option to patch, you can make new.txt old.txt again. Additionally, the patch utility can detect whether the file being patched already contains the patch you are applying. In this case, patch will ask you whether you want it to reverse or attempt to apply the patch anyway. Finally, the patch could fail. If this happens, a file named old.txt.rej (or something similar) will be created, and patch will exit. At that point, it is up to you to look at the contents of the .rej file (which will be in a patch format) and manually apply the contents to the source. (I have occasionally gotten a reject file to apply by using a larger **--fuzz** value of patch, but this can lead to patch application errors and subtle bugs that you'll be scratching your head about later.) Once the problems are worked through, however, you will have effectively merged two sets of changes into one.

Obviously, these examples are a bit contrived. In real-world practice, patch files are generally not applied by the same people who made them. Instead, you will

probably be either a provider of a specific patch (a source maintainer) or one who applies a specific patch (an end user).

Using patch-kernel

Although somewhat beyond the scope of this document, the Linux kernel actually includes a script which will aid you in keeping it up to date with the latest revisions. The **patch-kernel** script is located in the /usr/src/linux/scripts directory and will apply, in order, all the patches necessary to bring your kernel up to the latest revision, provided you have already downloaded them and put them in that directory. If you don't actually intend to participate in the Linux development effort, but just want to keep up with the latest and greatest source, this handy script will allow you to bypass the real workings of patch until you start developing. Once you start adding your own changes to your source tree, I recommend you use the manual method.

Maintenance of Source Trees

Getting back to the point of using diff and cousins as a sort of bare-bones revision-control system, I should mention there is obviously no one right or wrong way to maintain separate branches of a source tree. Low-level tools like these provide you with the framework to define the way you work, rather than forcing you into using one method or another. My suggestions here are only suggestions, and have been useful to me in the past when writing patches for open-source projects. However, I haven't exactly clocked the man-hours of Alan Cox working on the kernel or any other project, so there may be a better way. Please feel free to e-mail me at the address listed below with your thoughts.

When dealing with source trees, my general rule is always to maintain more than one. Unlike CVS or other revision-control systems, there is often no going back when you make a mistake. It is very easy to add instability to a stable patch, and no easy way to roll back your changes to a “last good” configuration. For example, let's imagine I have a source tree for a hypothetical project called “Foogram”. If I were maintaining this with CVS, I would need only one tree and a CVS server (which obviously maintains multiple trees). Since we don't have the advantage of a separate server with the bare-metal approach, I would generally have at least two directories for the project: foogram and foogram-last. (These are my personal naming conventions.) The last-released version would be in foogram, it's known to be stable and it's what I have to generate all my patches against when I want to distribute them. The foogram-last directory would contain my latest changes. This two-tiered approach is often effective enough for general use.

However, many projects actually get too complicated for this approach. I have been known to create a third (or fourth or fifth) directory which contains the

latest changes relative to foogram-last. For our purposes, I'll call this directory foogram-work. It often contains unstable and recent changes I've made to my source trees, which I want to keep separate from my more stable patches. It's important to keep a baseline directory to generate patches against, but this tends to lead to a rapidly expanding number of directories that other directories are relative against, and a mess for patching the manual way. In this example, I would try to maintain foogram-last as the baseline for the foogram-work directory by making sure I merge forward stable changes as I make them. If this were to become over-complicated, I would have to create a copy of foogram-last, called foogram-stable, which contained the copy of foogram-last that foogram-work was drawn against, so I could use created patches between the last and the stable directory to apply against the working directory in order to keep it up to date.

Confused yet? I sure am! Obviously, that's an extreme example, and many programmers will naturally simplify the process based on their own needs. Many projects do not require that level of complexity from individual developers. If you are reading this and still getting anything useful from it, you probably won't need to go to that complicated extreme to develop your own open-source modification patches, and instead will use the easy method.

The easy method is how it's possible to get away with keeping only one tree around, and this has worked for me on nearly every light project where I needed to modify only one or two source files as part of my patch. In that case, I recommend just making a copy of the source files you are editing, with a different extension (such as .old); and in the main tree, creating a diff that way using individual file patches. These small patches can be concatenated together when distributed to project maintainers. When doing it this way, you should be careful to run diff from the root directory of the source tree, so that the patch will be able to figure out later where the changed files were, especially if they were in different directories.

Conclusion

Much of the information here can be extracted from info pages and common sense. However, I hope that by documenting my own experiences with open-source projects and patches, I can encourage that small percentage of you who have the skill and desire to program the kernel, but have not chosen to do so. In the future, I hope to cover some of the other facets of kernel development that may be turning developers away, and I would be interested in hearing the reasons you might be nervous about lending your brain to some of these projects. Until then, happy hacking.

email: jpranevich@lycos.com

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

Archive Index Issue Table of Contents

Advanced search

Intel's Itanium on Launch Pad

Linley Gwennap

Issue #74, June 2000

The new Intel chip promises to take the PC to the high-end server market. Will Linux go along?

After more than six years of effort, Intel is finally ready to deploy its first processor based on the IA-64 instruction set, which it co-developed with Hewlett-Packard. The chip, named Itanium, is designed to go where no Pentium has gone before: the high end of the server market. When it arrives, Itanium will open doors not just for Intel but also for Linux. Today, a few large companies ship the vast majority of big servers. Sun, HP, IBM and Compaq servers rely mainly on proprietary RISC processors and proprietary versions of UNIX. But with the turbo-charged Itanium, many other vendors will deliver servers that match the power of these proprietary machines. These vendors are likely to rely on Linux or one of two key competitors: Monterey UNIX or Windows 2000.

Itanium Designed for Servers

Intel is no stranger to the server market. Its chips have been used in PC servers for years, and in 1998, the company introduced the Xeon product line specifically for bigger systems. But there is little to distinguish the latest Xeon products from the Pentium III chips in PCs: they use the same CPU, have the same bus bandwidth and remain limited to 32-bit virtual addressing. Itanium is Intel's first CPU designed from the ground up for server applications. For starters, the chip offers 64-bit addressing, providing programmers with much more flexibility when dealing with large objects and data sets. RISC processors have included 64-bit addressing for nearly a decade.

Instead of simply extending the aging x86 instruction set to 64 bits, Intel decided to leap to a new instruction set, IA-64. This state-of-the-art design includes a host of performance-enhancing features—such as predication, speculation and register frames—that are not found in x86 or RISC processors.

The general intent of these features is to give the compiler more control over the hardware. In theory, the compiler can make better decisions about how to group and order instructions for optimal performance. Furthermore, the processor hardware is simplified by relying on the compiler to make these decisions. Simpler processors can be both faster and less expensive to manufacture. Many previous instruction sets have failed to gain significant market share. The problem has been incompatibility with existing software. Itanium solves this problem by including an x86 translation unit on the chip. This unit can process any existing x86 application, albeit at a lower speed than an application that has been recompiled for IA-64.

Itanium Systems Nearly Ready

Intel has already produced hundreds of Itanium prototypes that are being used to validate new IA-64 hardware and software. Although there may be a few glitches remaining, the project appears on track. More than two dozen vendors plan to deploy Itanium systems, with the first systems appearing this fall.

In fact, nearly every major server vendor except Sun Microsystems has endorsed IA-64. Over time, HP will convert its entire line of RISC systems to the new architecture, as will SGI. IBM and Compaq will continue to develop RISC processors and systems, but will also deploy a broad range of IA-64 products. Companies such as Acer, Dell and Gateway will add Itanium to their Xeon server lines. With all of this backing, IA-64 is likely to grab at least 60% of the server market by 2003, according to a new report “Intel's Itanium and IA-64” from MicroDesign Resources. This would represent a big change from today's market, which is split among a variety of platforms. Linux holds 25% of the server market, according to IDC, a number that is rising rapidly. Although Linux is offered on several RISC platforms, the makers of these systems prefer to sell their own proprietary UNIX versions. Linux has been most popular on x86 systems from PC server makers. These vendors are most likely to benefit from Itanium. Current Intel processors can't match the performance and bandwidth of the best RISC systems, limiting PC server makers to low-end and mid-range systems. Using Itanium and Intel's new Lion motherboard, PC server makers will push into high-end markets, taking Linux with them.

Linux Competitors Join Forces

Linux faces strong competitors in the Itanium market. IBM and SCO are merging their versions of UNIX, AIX and UnixWare into an IA-64 operating system known as Monterey. Based on the AIX kernel, Monterey offers reliability and scalability features from IBM's high-end servers. Because of the popularity of UnixWare, Monterey is a natural choice for Itanium systems from Compaq, Dell and, of course, IBM. Microsoft hopes to dominate the IA-64 market with a 64-bit version of its Windows 2000 operating system. Most PC server vendors

will offer this as the default choice for Itanium systems. Previous Microsoft server products lack robustness and do not scale well beyond four processors; the company is working to fix these problems, but how long this will take remains to be seen. For small and mid-range servers, however, Windows will no doubt be a popular choice.

Linux's open-source model provides an advantage over these competitors, and it continues to gain features that will help it better compete against the likes of Monterey. As IA-64 processors displace proprietary RISCs from the high-end market, they create opportunities for newer operating systems. This platform change should boost Linux's prospects.

Linley Gwennap (linley@linleygroup.com) is the founder and principal analyst of The Linley Group (http://www.linleygroup.com/), a technology analysis firm in Mt. View, California, and the author of the report “Intel's Itanium and IA-64”.

Archive Index Issue Table of Contents

Advanced search

I'll Have My People Call Your People

Marcel Gagné

Issue #74, June 2000

Building a web-based telephone book can be easy, as long as you don't drink too much of Marcel's wine while you work.

Bon soir, mes amis. It is so very good to see you again. What? No, François, I do not think this is a good time. We have guests. Take a message and I will call later. Forgive me, mes amis. François has been run off his feet since we started distributing our new on-line phone book. My impetuous waiter put our name as a default record in each of the databases, and now everyone is calling! Mon Dieu!

The phone book? But, of course. In fact, in honor of this month's issue celebrating “The People Behind Linux”, we are cooking up a web-based telephone book made with rich, yet low-calorie, open-source software and our secret ingredient, Linux. I think you will find the recipe quite enjoyable and practical too, non? What is the point of knowing all these people if you never talk to them? That is why François' mother insisted that her name be put in. I jest. Please sit, mes amis, while I show you how to create your own web-based telephone book, which is sure to become a centerpiece of your intranet. For added spice, this wonder of the intranet even provides an IN/OUT board.

François! Bring some wine for our guests. Vite!

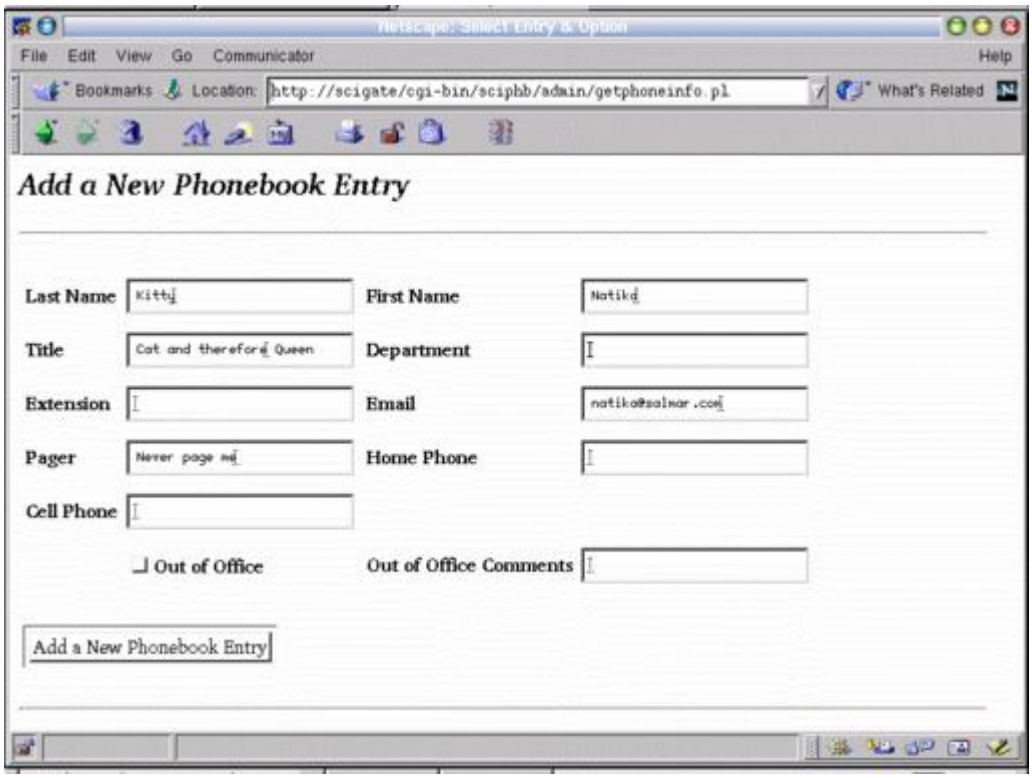


Figure 1. The web phonebook's search screen

Before we get into the *meat* of this recipe, may I suggest that you look at Figure 1 which shows the administration search screen. The standard user's screen is similar, but does not have the option to **add** a user. Figure 2 shows the results of a fairly wide search. Note the gray and red buttons that allow you to modify or delete an entry.

Figure 2. Administration Search Results Screen

Our telephone book is written entirely in Perl and uses PostgreSQL to store its entries. Before I get too far, I will tell you what you will need,

- PostgreSQL
- DBD and DBI modules for PostgreSQL from CPAN
- The cgi-lib.pl Perl library

In all likelihood, Perl is already loaded on your system. Most distributions include it as part of the standard install. You should also have the Apache web server running. Again, it is extremely likely that you installed Apache with your system.

Preparation

PostgreSQL can be found at the official PostgreSQL site and, in most cases, on your Linux distribution CD. For my Red Hat system, it was simply a matter of installing PostgreSQL with the **rpm** command.

```
rpm -ivh postgresql-6.5.2-1.i386.rpm
rpm -ivh postgresql-devel-6.5.2-1.i386.rpm
rpm -ivh postgresql-server.6.5.2-1.i386.rpm
```

Depending on when you obtain and load PostgreSQL, the numbers may vary. Part of what happens during this installation is to create a **postgres** user. After assigning a password to this user, log in as postgres and initialize the database environment. Remember that the paths may vary depending on your installation.

```
initdb --pglib=/usr/lib/pgsql --pgdata=/var/lib/pgsql
```

Next, you will need to create some default PostgreSQL users. If you are installing as root (for access to Perl directories, cgi directories, etc.), then root will have to be added. So will the user “nobody”. This is often the user of your httpd server. Some have a user called “www” to run web services. I will use “nobody” as mine, and you may use whatever your server is configured for. Start by logging in as your postgres user, and execute the following command to add the users root and nobody to your PostgreSQL system:

```
createuser root
```

You'll be prompted for root's UID (accept the default) and whether user root is allowed to create databases. Answer “y”. When asked whether root is allowed to create users, I answered “n”. Now, do the same thing for user nobody. The only difference is that I answer “n” to the question of whether nobody is allowed to create databases as well. Depending on which version of PostgreSQL you are using, the question of whether a user is allowed to create other users may be worded this way:

```
Is user "whoever" a superuser?
```

The answer is still n, or “no”. Finally, with the creation of nobody, you will then be asked whether **createuser** should create a database for nobody. Answer “y” and you are finished creating users.

Both the **DBD** and **DBI** module can be found at the CPAN FTP site, a huge Perl resource on the web. Since the two modules are in slightly different directories, I will give you both: ftp://ftp.cpan.org/CPAN/modules/by-module/DBI/ and ftp://ftp.cpan.org/CPAN/modules/by-module/DBD/.

At the time of this writing, the latest and greatest DBI version was DBI-1.13.tar.gz, whereas DBD weighed in at a comfortable DPD-Pg-0.93.tar.gz.

The DBI module is common to all the various databases, but the DBD module must be specific. DBD modules are available for numerous databases such as Oracle, Informix and, of course, PostgreSQL. DBI is an application program interface (API) for Perl5 to interface with database systems. The idea is to provide a consistent set of modules and calls so that database access code is portable without too much fuss. Since each database will vary somewhat, however, the DBD module comes into play to take those differences into consideration.

For more information on how DBI works, try this address: www.isc.org/services/public/lists/dbi-lists.html. You do not need the information in order to be able to cook up today's recipe, but it makes for interesting reading later.

Install the DBI module first by unpacking the distributions into some temporary directory and following these steps:

```
cd /usr/local/temp_dir
tar -xzvf DBI-1.13.tar.gz
cd DBI-1.13
perl Makefile.PL
make
make test
make install
```

Tres simple. To install the DBD module, the process is similar. The latest version of the DBD module now requires you to set a couple of environment variables before the install can occur. These are **POSTGRES_LIB** and **POSTGRES_INCLUDE**.

```
POSTGRES_LIB=/usr/lib/pgsql
export POSTGRES_LIB
POSTGRES_INCLUDE=/usr/include/pgsql
export POSTGRES_INCLUDE
```

Now, you can run the install:

```
cd /usr/local/temp_dir
tar -xzvf DBD-Pg-0.93.tar.gz
```

```
cd DBD-Pg-0.93
perl Makefile.PL
make
make test
make install
```

The final ingredient you will need is the cgi-lib.pl library for Perl. This has become a virtual (de facto) standard for CGI design using forms. Surf over to the following address and save the file into your /usr/lib/perl5 directory: http://cgi-lib.berkeley.edu/.

If your Perl libraries live in a different directory (e.g., /usr/local/lib/perl5), you will need to modify the **require /usr/lib/perl5/cgi-lib.pl** line near the top of each cgi-bin perl script to reflect your own directory structure.

Cooking Instructions (Installation)

After all these steps are completed, we have pretty much all the pieces we need to continue with the actual phone book creation and build. If you haven't already done so, obtain the distribution for the phone book from the Salmar web site in the downloads section or from the *Linux Journal* FTP web site (see Resources).

The tar, gzipped distribution file is designed to extract into the standard Red Hat file structure for the Apache server, namely /home/httpd/html and /home/http/cgi-bin.

```
cd /home/httpd
tar -xzvf /path_to/phonebook.tar.gz
```

The file list looks something like this:

```
html/sciphb/
html/sciphb/index.html
html/sciphb/admin/
html/sciphb/admin/index.html
cgi-bin/sciphb/
cgi-bin/sciphb/admin/
cgi-bin/sciphb/admin/getphoneinfo.pl
cgi-bin/sciphb/admin/updphone.pl
cgi-bin/sciphb/admin/getphone.pl
cgi-bin/sciphb/admin/createphdbs.pl
cgi-bin/sciphb/ugetphone.pl
cgi-bin/sciphb/getnextkey.pl
```

If you are running Apache from binaries built from the default source distribution, those directories will likely be /usr/local/apache/htdocs and /usr/local/apache/cgi-bin. If this is the case, you can extract the files to a temporary directory and move the sciphb directories to the appropriate cgi-bin and html/htdocs. Since there is always more than one way to do it, you could create symbolic links like this:

```
mkdir /home/httpd
ln -s /usr/local/apache/htdocs /home/httpd/html
ln -s /usr/local/apache/cgi-bin /home/httpd/cgi-bin
```

Everything will work fine from here, but you may want to verify the path to Perl. The .pl files in the cgi-bin/sciphb directory all call Perl from /usr/bin/perl. If necessary, change the first line of these files to reflect the actual path to your Perl binary.

Let this simmer for a few seconds, then change directory to the admin cgi scripts location:

```
cd /home/httpd/cgi-bin/sciphb/admin
```

There you will find a file called **createphdbs.pl** which (strangely enough) will create the databases necessary to use the phone book. This package is quite simple and, I believe, will provide the aspiring chef plenty of inspiration for further development. The scripts are small in number, but going through each script would take far too much space and take great advantage of the hospitality of this fine publication. I would, however, like to show you the PostgreSQL table creation for this on-line directory. Keep in mind that this is just a code morsel (Listing 1) and not the finished product. The first section shows the setup and initial calls using the Perl DBI in order to create the phone database. In particular, pay close attention to the line that begins with

```
$dbh = DBI->connect("dbi:Pg:dbname=$dbmain",
```

The dbi:Pg identifies the database interface type (DBD); in this case, **Pg** stands for PostgreSQL.

Listing 1

Listing 2 is a portion of the actual database creation. Note the **$dbh** calls throughout. These are what make the DBI environment portable. Once having established the DBD (as in the connect statement above), there should be very little need to modify this code if you were to use another database.

Listing 2

Of course, there is more, but you will have to download the scripts to satisfy your appetite.

Serving Suggestions: Your Completed Phone Book

The phone book has two different interfaces: one for lookups only and another for both lookups and updates. You can, if you wish, use only the administration page if you are content to allow anyone in your organization to update the phone book information. Modifications of the index.html file to pull in the administration page require only a change to the actual page being pulled in.

To access the default user interface, use this link: `http://your_server/sciphb/`.

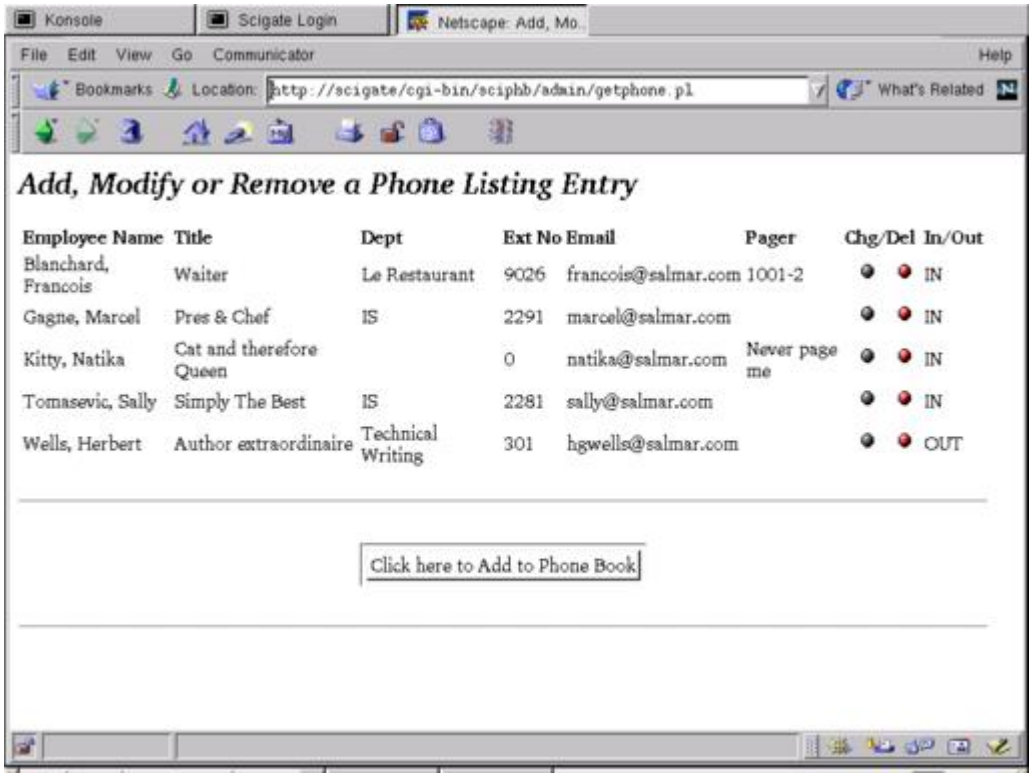


Figure 3. Adding Users Through the Phonebook's Administration

The administration version of this page (which allows the adding of users, modification of records and deletes) can be accessed by going to this address: `http://your_server/sciphb/admin/`.

I recommend you keep the user and administration portions separate. Give a few specific users the responsibility of updating the phone book, and the information should continue to make sense.

You might also consider changing the available fields. For instance, the IN/OUT board aspect of the directory may not be for you. Perhaps you need another field. With the scripts provided and a little time, modifying this telephone directory to suit specific needs should be quite simple.

Once again, mes amis, it is closing time. The pleasure of having you here on a regular basis puts, how shall I say, “pressure” on my wine cellar. Ho, ho! I shall have to replenish, non? It is okay, mes amis. Chef Marcel is only kidding. There is always more wine, but the doors, they must close sometime. Until next time, enjoy that on-line telephone book. Remember—you are always welcome here at Chez Marcel. Bon Appétit!

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. He loves Linux and all flavors of UNIX, and will even admit it in public.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Building Sites with Mason

Reuven M. Lerner

Issue #74, June 2000

This month, Mr. Lerner introduces us to a Mod-perl module to aid in building large, dynamic web sites.

When we speak of dynamically generated web content, most people immediately think of CGI, the “common gateway interface”. CGI is portable across web servers, languages and operating systems, but that portability comes at the expense of efficiency. For example, every invocation of a CGI program written in Perl results in the creation of a new process, in which the program must be compiled and then interpreted.

Programmers willing to trade speed for portability have a number of options at their disposal. Many Perl programmers have chosen to use **mod_perl**, which makes it possible to modify Apache's behavior using Perl modules. A Perl module invoked by mod_perl is run inside the Apache process, removing the overhead associated with starting a separate CGI process. **mod_perl** also caches the compiled version of invoked modules, eliminating the need to compile them each time they are run. The result is a dramatic improvement in speed, as well as the flexibility to modify Apache quickly and easily using Perl.

For all its power, mod_perl has never appealed to me for creating large, dynamic web sites. True, the increase in speed is extremely impressive, and it is not that hard to work with. As we have seen in several previous editions of this column, writing a mod_perl module can be quite easy, and it integrates smoothly into a larger web site.

When a site needs to create a large quantity of dynamic output, much of which is written and designed by non-programmers, mod_perl's power is hampered by the need to use dozens or hundreds of modules, each servicing only a single URL or directory. The solution to this problem is to integrate mod_perl with templates, which intersperse HTML-formatted text with Perl code. We have

looked at templates on several previous occasions and have seen their power and flexibility.

This month, we will look at Mason, a mod_perl module written by Jonathan Swartz, which attempts to solve many of these problems. It uses templates and encourages the use of separate "components", which can be built up to create a large, dynamically generated site. Because these components exist in separate files, Mason offers additional advantages:

- It caches components, providing a bigger speed boost than that from simple templates.
- It provides a complete debugging and previewing environment.
- It produces output files.

I first heard about Mason nearly two years ago, and kept telling myself I would look at it one day. I finally took a serious look at Mason several months ago, and was extremely impressed with what I saw—enough that I expect to do most of my future development with it.

The number of publicly distributed Mason components is still relatively small, which makes it seem like a poorer development environment than Zope and commercial template solutions. However, this situation already appears to be changing, and the number of components will most likely increase significantly during the coming months and years.

Installing and Configuring Mason

While Mason can work as a CGI program, it works best and most easily with mod_perl. (See Resources for information on obtaining and installing Apache and mod_perl.) Retrieve the latest version of **HTML::Mason** from CPAN and follow the same procedure as you would for any other module, detailed in the INSTALL file that comes with it.

However, using Mason is more complicated than simply saying "use HTML::Mason". Because it works via mod_perl, which is part of Apache, the Mason configuration must be performed when Apache starts up. This is accomplished most easily by using a **PerlRequire** statement in the Apache configuration file, normally called httpd.conf:

```
PerlRequire /usr/local/apache/conf/mason.pl
```

The above statement tells Apache to execute the Perl statements in /usr/local/apache/conf/mason.pl when it first starts up. Any Perl modules imported or variables declared in mason.pl are placed into a section of memory shared across all Apache processes. This can mean a substantial memory savings,

since Perl and mod_perl consume a large amount of memory, and most web servers run prefork child Apache processes.

At the very least, **mason.pl** must create three different objects:

- The Mason parser (**$parser**), which turns each Mason component into a Perl subroutine.
- The Mason interpreter (**$interp**), which executes the subroutines that were created by the parser. When creating the interpreter (**HTML::Mason::Interp**), we name two directories in which the interpreter reads and writes information: the Mason "component root" where the Mason components sit, and the Mason "data directory" in which caching and debugging information are stored. I normally set the component root to /usr/local/apache/mason and the data directory to /usr/local/apache/masondata.
- The Mason ApacheHandler (**$ah**), which handles the HTTP request and generates a response.

Warning: for reasons that are not entirely clear, Mason cannot handle symbolic links. Specifying a symbolic link as a directory name will lead to mysterious "File not found" errors. On my system, /usr/local is a symbolic link to /quam/local; while the Mason documentation does mention this quirk, it was not explicit enough to save me more than half an hour of installation.

Listing 1

A bare-bones mason.pl is shown in Listing 1. Notice how mason.pl defines the subroutine **HTML::Mason::handler**, which is invoked once for each incoming HTTP request. In this way, Mason is able to handle each HTTP request; the ApacheHandler takes the request and hands it to the Interpreter, which then reads a compiled component from the cache or parses it as necessary.

More advanced Mason installations use mason.pl to define all sorts of additional behavior. For example, Mason comes with a previewer/debugger component, making it possible to trace through the execution of a component and its subcomponents. It is also possible to define different ApacheHandler objects, one for each type of browser or request type.

Once our mason.pl file is installed, we must tell Apache to let HTML::Mason handle some incoming requests, rather than using the default Apache handlers. This is where we connect the component root to the Apache Handler. For example, if the component root is /usr/local/apache/mason, we can say the following:

```
Alias /mason /usr/local/apache/mason
<Location /mason>
    SetHandler perl-script
    PerlHandler HTML::Mason
</Location>
```

The Alias directive tells Apache to translate every URL beginning with /mason to the Mason component root, /usr/local/apache/mason. The <Location> section tells Apache that every URL beginning with /mason should then be handled by the mod_perl handler HTML::Mason.

Mason Components

In the Mason universe, a "component" can return either HTML or a value. The former usually consists of HTML templates or template fragments, while the latter consists of subroutines and other code which are invoked by templates. All components share the same syntax, which should be familiar to anyone who has used a template system.

Perl code can be placed inside a component, bracketed by **<%** and **%>**. Any returned value is inserted into the component, replacing the Perl code that created it. For example, the following component (output.html) will display the current time of day each time it is invoked:

```
<HTML>
<Head><Title>Current time</Title></Head>
<Body>
<H1>Current time</H1>
<P>The current time is: <% scalar localtime %></P>
</Body>
</HTML>
```

I put the above into the file time.html and placed it in the component root directory. Immediately after doing so, I was able to go to the URL /mason/time.html and get the current time.

Mason supports two other types of Perl sections, which can be useful in different contexts. A **%** in the first column of a Mason component forces the entire line to be interpreted as Perl, rather than literally. This is best used for control structures (such as loops and if-then statements) that produce text strings, as in the following:

```
<HTML>
<Head><Title>Current time</Title></Head>
<Body>
<H1>Months</H1>
% foreach my $month (qw(Jan Feb Mar Apr May Jun
% Jul Aug Sep Oct Nov Dec))
% {
<P><% $month %></P>
% }
</Body>
</HTML>
```

As you can see, the **<% %>** construct works in all contexts. In addition, lexicals declared at the top level of one Perl segment can be used within any other Perl segment.

Finally, long runs of Perl can be placed inside **%perl** blocks. This is best for doing heavy-duty computation, rather than simply retrieving variable values. For example:

```
<HTML>
<Head><Title>Current time</Title></Head>
<Body>
<H1>Months</H1>
<%perl>
my @months = qw(January February March April May
June July August September October November December);
</%perl>
<P>The current month is <% $months[(localtime)[4]]
%>.</P>
</Body>
</HTML>
```

Once again, notice how the lexical (**my**) variable declared in the **<%perl>** section is available in the following **<% %>** section.

Calling Components

Experienced users of **Text::Template** and other Perl templating modules are probably not very impressed at this point. After all, there are dozens of ways to create templates of this sort, and many work with mod_perl for extra speed.

However, Mason's template syntax includes provisions for invoking other components, much as one subroutine might invoke another. (Indeed, since the Mason parser turns each component into a subroutine, this is not an incorrect analogy.) In some ways, this is like having a heavy-duty server-side include system, allowing you to standardize headers and footers. However, because components can return values as well as HTML output, and because Mason makes it possible to pass arguments to a component, things can get far more interesting.

One component can invoke another component with the special **<& &>** syntax. For example, the following invokes the component **subcomp**:

```
<& subcomp &>
```

Any HTML produced by subcomp is placed at the point where it was invoked, much like a server-side include. Each HTML page generated by a Mason site can consist of one, five, 10, 20 or more components. In this way, it is possible to assemble a page from individual elements—beginning with headers and footers and moving on to tables and pull-down menus. For example, here is a header component:

```
<!-- begin component: header.comp -->
<Body bgcolor="#FFFFFF">
<H1>This is a header</H1>
<!-- end component: header.comp -->
```

And here is a footer component:

```
<!-- begin component: footer.comp -->
<address>
<a href="mailto:reuven@lerner.co.il">
reuven@lerner.co.il</a>
</address>
<!-- end component: footer.comp -->
```

Finally, here is a top-level component in which the header and footer come from the above components:

```
<HTML>
<Head><Title>Title</Title></Head>
<& header.comp &>
<P>This is the body</P>
<& footer.comp &>
</HTML>
```

Notice, I gave file extensions of “comp” rather than “html” to the header and footer. This is simply a convention that enables me to differentiate between top-level components (which have .html extensions) and lower-level fragments.

Also, notice how I begin and end each lower-level component with HTML comments that indicate where it begins and ends. This provides a primitive type of debugging (expanded by the Mason previewer/debugger component) that lets me see where things are happening, simply by viewing a component's HTML source code.

Arguments

The above examples of header and footer components are good for simple sites. However, it would be more useful if our header and footer components could take arguments, allowing us to modify parts of their content as necessary.

Mason indeed allows components to send and receive arguments, giving an extra level of flexibility. To pass arguments to an invoked component, place a comma between the component's name and a list of name,value pairs. For example:

```
<& header, "address" => 'president@whitehouse.gov' &>
```

Components can receive passed arguments in special **<%args>** sections, traditionally placed at the bottom of a component file. An **<%args>** section declares arguments for a component, with an optional default value if none are passed to the component. For example, the following **<%args>** section declares the **$name** and **$address** variables. An argument without a default variable is

mandatory. **$name** has no default value, while **$address** has a default value of reuven@lerner.co.il:

```
<%args>
$name
$address => 'reuven@lerner.co.il'
</%args>
```

We can rewrite footer.comp in this way:

```
<!-- begin component: footer.comp -->
<address>
<a href="<% $address %>"><% $name ? $name : $address %></a>
</address>
<%args>
$name => ""
$address => 'reuven@lerner.co.il'
</%args>
<!-- end component: footer.comp -->
```

Finally, we can rewrite output.html to send the required parameter without the optional parameter:

```
<HTML>
<Head><Title>Title</Title></Head>
<& header.comp &>
<P>This is the body</P>
<& footer.comp, "name" => 'Reuven' &>
</HTML>
```

$m and $r

Experienced mod_perl programmers might like the idea of the components Mason provides. However, there are times when it is easiest to accomplish something by reaching into the guts of Apache and working with the mod_perl request object, traditionally called **$r**.

Mason provides each component with a copy of **$r**, so we can work with the internals of the server. For example, we can send an HTTP Content-type of "text/html" by using the **content_type** method:

```
$r->content_type("text/html");
```

Because **<%perl>** sections are invoked before the actual HTTP headers are returned, Mason components can modify all response headers in this way, including working with HTTP cookies.

A similar object, called **$m**, is specific to Mason and allows us to invoke methods having to do with Mason components and development. For example, we can retrieve the contents of a component with the **$m->scomp** method. The manual page at **HTML::Mason::Devel** lists many more methods that can be invoked on **$m**.

Initialization

Mason gives us two sections, **<%init>** and **<%once>**, in which to run Perl code at the beginning of a component's execution.

An **<%init>** section is evaluated before any **<%perl>** sections, as well as any other Perl code in the component. This gives the component a chance to define variables and retrieve information on its environment. In effect, **<%init>** is the same as **<%perl>**, except it can be placed anywhere in the component, rather than at the top. Traditionally, **<%init>** sections are placed near the bottom, along with the other special sections.

An **<%init>** section is evaluated each time a component is invoked. However, there are items that need to be created only the first time a component is invoked, rather than every time. Such items can be put in a **<%once>** section. Lexicals and subroutines declared within a **<%once>** section remain throughout the life of the component, making them particularly useful for initializing the component's state. However, **<%once>** sections are not evaluated within the context of a Mason request, which means they cannot invoke other components.

Mason components that connect to a relational database with Perl's DBI often use a combination of **<%once>**, **<%init>** and **$m** to reuse database handles. We can do the following, for example, as suggested in the Mason documentation:

```
<%once>
my $dbh;    # Declare $dbh only once
</%once>
<%init>
# If this is the first time we're running,
# connect to the database
if ($m->current_comp->first_time)
{
$dbh = DBI->connect("DBI:mysql:$database:localhost",
                 $username, $password) ||
die qq{DBI error from connect: "$DBI::errstr"};
}
</init>
```

autohandler and dhandler

While Mason components can create headers and footers using the **<& &>** syntax we saw above, it becomes cumbersome to put such sections inside each top-level component we create. For this reason, Mason supports two special kinds of components, one called **autohandler** and the other **dhandler**.

If an autohandler component exists, it is invoked before each component in the directory. That is, the autohandler is invoked and can produce HTML output of its own before retrieving the component that was actually requested, with **$m-**

>call_next. For example, the following autohandler will put a uniform title and footer on each document in its directory:

```
<HTML>
<Head><Title>Welcome to our
site!</Title></Head>
<Body>
<% $m->call_next %>
<hr>
<address>webmaster@example.com</address>
</Body>
</HTML>
```

dhandler, by contrast, is invoked if a component does not exist. In some ways, this allows us to rewrite the “404--No such file” error message that web sites often produce.

While autohandlers normally influence only their own directories, dhandlers affect all subdirectories. Thus, a dhandler in /foo will affect all documents in /foo/bar, but not in /bar. However, an autohandler in /foo will not affect items in either /foo/bar or /bar.

Slide Show

Now that we have seen how Mason can work for some simple tasks, let's look at some components I wrote for creating slide shows. Such presentations will not have the fancy wipes and graphics available with Microsoft's PowerPoint, but are more than adequate for most technically oriented groups.

The slide show component consists of an autohandler, a dhandler and one or more slides (text files) written in HTML. Each slide consists of a piece of HTML that will be stuck inside the **<Body>**. For example, the following could be a slide:

```
<H1>Short Presentation</H1>
<P>This is my short presentation.</P>
```

Inside the autohandler (Listing 2; see Resources) we have a **<%once>** section that defines several constants we will reuse, as well as **@slides**, an array containing the list of slides. For example, here is the value of **@slides** from a talk I recently gave:

```
my @slides = qw(start whoami free-software just-in-time
  databases mysql postgresql
  cgi mod_perl
  templates text::template minivend minivend-example
  mason mason-example mason-autohandler
  php jsp zope acs xml
  conclusion);
```

By reordering the file names within **@slides**, I change the order of my presentation, and by removing or adding elements from **@slides**, I can change the length of the presentation.

The autohandler uses **$m->scomp**, described earlier, to retrieve the HTML associated with a slide. It uses this to retrieve any headline (in **<H1>** tags) it might find within the slide and uses the headline in the **<Title>** tag.

In addition, the autohandler produces links for the “previous” and “next” slides. We do this by getting the index of the current slide and retrieving the names from the array:

```
my $previous_slide = $slides[$current_slide_index - 1] || $slides[0];
my $next_slide = $slides[$current_slide_index + 1] || $slides[0];
```

Once we have the names of the previous and next slides, we can retrieve their headlines, making for attractive “previous” and “next” links:

```
# Grab the headline from the next component
my $next_headline = $next_slide;
my $next_contents = ($m->scomp($next_slide));
if ($next_contents =~ m|<H1>(.+)</H1>|igs)
{
    $next_headline = $1;
}
```

One of the nice things about using this autohandler for slides is that it allows me to reorder or modify a talk by shifting the names of the files.

In addition to the autohandler, I installed a dhandler to take care of mistaken filenames:

```
<HTML>
<Head><Title>Error: No such
page</Title></Head>
<Body BGCOLOR="#FFFFFF">
<P>Sorry, but the page <i><% $r->filename() %></i> does not exist.</P>
</Body>
</HTML>
```

Conclusion

Mason provides an environment balanced nicely between simple, easy-to-use templates and the complex, powerful underpinnings of mod_perl. If you ever considered using mod_perl on your site, but were scared away by the complexity, consider looking into Mason. Not only is Mason free software—a good thing, for a variety of reasons—but it is a proven tool that makes web development significantly easier than many of its counterparts. I hope to do much development in Mason over the coming months, and hope to share many of my experiences and code as I grow to enjoy this new tool.

Resources

Reuven M. Lerner , an Internet and Web consultant, recently moved to Modi'in, Israel following his marriage to Shira Friedman-Lerner. 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

The Penguin and the Dinosaur

Adam J. Thornton

Issue #74, June 2000

Think Linux is only for the PC? Think again.

Although Linux began its life on the Intel x86 architecture and still has strong roots there, it is highly portable. Linux runs on everything from SGIs at the high end to Palm Pilots, Psion PDAs and tiny embedded microcontrollers at the low end.

Linux's world just got a little larger. It now runs on the IBM System/390 mainframe. That's right; there is now a port of Linux to the IBM System/390 mainframe architecture. Once you've booted it, it works much like you'd expect any other Linux system to. As a matter of fact, there are *two* ports: there's IBM's, which runs only on relatively recent System/390 machines and was developed in secret, and there's also Linas Vepstas' “Bigfoot” port, which runs on older System/370 machines as well and was done as a proper open-source project. Bigfoot was quite close to running—it boots the kernel and loads /bin/sh—when IBM dropped the bomb. Reconciliation of the two projects is an ongoing and somewhat acrimonious issue.

VM on the Mainframe

I can hear some readers scratching their heads in puzzlement. If you have a mainframe, you presumably already have a perfectly good and extremely expensive operating system for it, right? Running Linux on such a machine means taking down what is almost certainly an incredibly expensive and probably highly loaded production machine—not a good idea.

Well, the first part is true. However, if you have a recent S/390 *or* any version of VM (virtual machine), you don't have to interrupt service at all to play with other operating systems on your production machine. If you're coming to this from the Linux/x86 world, you're probably aware of VMWare, which effectively splits your PC and lets you boot Windows NT in a virtual machine under Linux (and if

you're not, you should be—it's an incredibly cool application). Well, IBM mainframes did it first.

The VM operating system (also called a “hypervisor”, like a supervisor on steroids) was designed from the start to do exactly that; in essence, under VM, each user gets his or her own copy of the machine. It looks exactly like a System/390, complete with attached peripherals, and the user owns (effectively, is root on) that virtual machine. Behind the scenes, of course, the peripherals are simulated and are managed by VM, and the hardware is designed with virtualization assistance features which VM exploits. This is analogous to the way VMWare provides “disk access” by simulating each disk as a file within the Linux file system. VM can also simply present an actual physical device, if appropriate to your needs.

This gets *really* interesting when you realize you can run VM in your own Virtual Machine. In fact, a very common use of VM is testing a new system second-level; that is, running underneath the instance of VM actually running on the bare metal. This allows you to make sure all your existing programs still work. Only when you have all the bugs worked out of the new system do you briefly shut down the machine and bring it back up with the new system running on the actual, rather than a virtual, machine. You can be confident it will work, since you've already extensively tested it.

There are other tricks you can do with VM. IBM's mainframe cash cow is OS/390, formerly MVS. You can run OS/390—or VSE, the third mainframe OS—in a virtual machine, although you cannot run guest operating systems under OS/390. Thus, VM lets you do the same test upgrade process with OS/390 as it does with VM itself. The virtual machines do not have to reflect the hardware configuration of the physical machine: virtual machines can be multiprocessor machines, even on a uniprocessor real system, or vice versa. Give each machine as much memory as you think it ought to have and its own set of printers, tapes, LAN adapters, whatever.

This is such a massively useful feature that IBM, in its more recent machines, has included a hardware feature which acts as a stripped-down VM in microcode; whether it *is*, in fact, a stripped-down older version of VM is a subject of debate. In any event, even without running VM, you get the benefit of LPARs (“Logical PARtitions”) which effectively allow you to partition your mainframe into a small number (usually 15 or fewer) of conceptual machines. IBM's mainframe competitors offer essentially the same functionality under different names, such as Amdahl's MDF.

How efficient is virtualization under VM? The architecture was designed with self-virtualization in mind, so it is much more efficient than VMWare. When a

virtual machine under VM isn't doing anything, it consumes very little in the way of resources. Medium to large VM systems typically support about 5000 simultaneous logins (each, mind you, with its own virtual machine) without hiccuping.

David Boyes at Dimension Enterprises recently did a web server torture test—in an LPAR running a single VM instance, so *his* machine was running second-level on a medium-sized S/390, and each of his Linux machines was running *third*-level. His one goal was to see how many Linux boxes he could bring up, simultaneously serving requests. The results were somewhat astounding.

Are you sitting down? He ran out of resources at 41,400 simultaneous Linux machines—forty-one thousand four hundred. I'll get to what *that's* good for a little later; the notion of 41,400 web servers on a single physical box ought to give you an idea, though.

The upshot is, if you have access to a recent System/390, you can almost certainly spare the resources to play with Linux/390 without noticeably affecting your production system. If you *don't* have access to a mainframe, well, just keep reading. I have a little surprise coming up.

Linux on the Mainframe

One obvious question is, “what good does this do?” Let's look at a few scenarios. Scott Courtney has addressed this issue wonderfully in his essay at LinuxPlanet (see Resources), which should be required reading if you're interested in L/390—it is a great deal more in-depth than my article.

UNIX administration and development skills are much more common and much cheaper than mainframe skills. Thus, it's going to be easier to find people to work on Linux on your mainframe than it would be to find OS/390, VSE or VM gurus.

What good is a mainframe? Mainframes traditionally had little in the way of CPU power (no longer true—you get plenty of raw CPU speed out of one), but have absolutely fantastic I/O capabilities. One of the main needs in a big web server farm or a big e-commerce server is I/O, of course.

Linux/390 presents an interesting migration path for organizations which are seeking to de-emphasize their mainframes but can't just decommission them, because it provides services simply unavailable in other environments. I've personally seen this scenario happen twice: once at Rice University and once at Princeton. Plans to shut the dinosaur down were announced, then retracted once it became clear that vital parts of the campus would grind to a halt, since

there was simply no viable alternative to some of the mainframe services. Moving over to Linux for those things it *can* do provides a smoother and cheaper transition, without the need for additional hardware. Let's face it: if your organization is going to be moving away from OS/390 or VM, better they should move to Linux than to another OS. Since Linux running Samba is already a good back-office substitute for NT, you could provide Windows browsing services to your users without ever needing PC hardware, let alone an NT license.

Another exciting possibility is that VM licenses tend to be held by academic institutions. Imagine a third-year computer science course on operating systems or networking. Now imagine each student getting his very own Linux box with which to play. There's full OS source code, a full development environment and isolation from the production systems and other students. A fantastic course could be developed around a study of Linux internals in such an environment, and a medium-sized S/390 could support a class of 25 students, all recompiling the kernel at once, without breaking a sweat.

The commercial version of this scenario is the commercial web-hosting server. Traditionally, this means you get dedicated access to a machine in a rack space somewhere, physically managed by an ISP. We'll do the numbers a little later, but in short, if you're doing this on a large scale, the price of a mainframe and a VM license get dwarfed quickly by the price of a whole bunch of fast, rack-mounted PCs. Your labor cost also drops radically, as you don't have to physically set up yet another PC; you simply create one more virtual machine on your VM box, and give it its own copy of the installed system disks.

As an aside, it doesn't hurt to remember that the total cost of network ownership typically breaks down to less than one-quarter hardware, roughly one-third service and facilities, with the remainder the necessary staff to support it (IDC, 1996). Administration tasks are obviously greatly simplified when the entire network of Linux machines is contained within a single box.

It would also be quite possible to separate various services onto various virtual machines. Sendmail would get its own (virtual) Linux box, DNS another, Samba another. This would be good from both a security standpoint (an exploit on one machine compromises only one service) and a reliability standpoint. You can also split the various pieces of a multi-tier application (e.g., web front end, business rules processing engine in the middle and RDBMS on the back end) among separate virtual machines, and run your database on OS/390, if you prefer. The isolation would make both debugging and development somewhat easier. I *know* this is something we always laugh at the NT people for requiring,

but there are three advantages to the Linux-on-a-virtual-machine method of service isolation:

- Additional hardware cost is zero, as opposed to a couple thousand dollars per machine.
- Additional software cost is zero, as opposed to the cost of an NT license per machine.
- Actual resource utilization overhead is very low, since VM's virtual machines consume almost no resources unless they are actually running.

Price and Performance

Mainframes are expensive. There's no way around that fact. They're less expensive than they used to be, but they're certainly not $800 PCs. I'm not particularly *au courant* with IBM's pricing structure, but let's take a million dollars as a high-end price. A million dollars will buy you a *lot* of mainframe; you'd get a terrific IBM support contract with it and a VM license, as well as a backup solution. Most mainframe shops run OS/390, often in tandem with VM, but if your purpose is to run many Linux virtual machines, then you'd want VM and would have no use for OS/390 unless you wanted to do traditional mainframe computing too. I'm told a brand-new, top-end system with several terabytes of DASD is closer to $600,000 than a million, and much cheaper secondhand. However, for the purposes of argument, let's stick with a million as a nice round figure.

What do you get for that price? A machine which will run—I'm being extremely conservative here—1000 simultaneous Linux machines without a problem. You're talking $1000 per Linux box: not too different from the cost of a low-end rackmount Linux system, thus the price right there is a wash.

Obviously, it's a lot cheaper to rent space for a single S/390 and its associated disk arrays than it is to rent space for 1000 physical Linux boxes; even in one-unit packaging at 42 units/rack, you're still talking 24 racks. That's certainly more than even a big S/390 is going to take. Networking gets easier too; buy the 155MBps ATM options on your OSA cards, plug up to twelve ATM interfaces into your machine (if you need more scalability, options exist for you) and coordinate internal communication between your virtual machines via a virtual LAN. An internal, virtual LAN is much easier and cleaner to administer than a physical topology of switches and runs of Cat5 and fiber.

You can cut back on resource usage, too. If some of your machines will have identical file systems (it's fairly common, for example, in the UNIX world, to mount /usr read-only and have symlinks to what needs to be writable), then you can maintain a single disk with the shared file system and mount it from

each of the virtual machines. It's just like sharing file systems over NFS, only without the ugliness of NFS. This, incidentally, was what David Boyes did when running his 41,400 machines: all shared /usr and /bin; with a little more trickery, /lib could be shared too. Furthermore, there's no need to specify much—or indeed, any—swap for your Linux machines; VM knows all about memory management and does it very effectively. Give each of your Linux machines a bunch of virtual memory, and let the VM hypervisor worry about paging it in and out.

Now let's look at reliability. We'll assume one of these $1000 machines has a MTBF of 1000 days. That's probably on the high side for a thousand-dollar machine, if you're pushing it fairly hard. At $1000, you're not getting RAID, your disks are probably IDE, and even redundant power supplies are unlikely. If you have a thousand of these boxes in a room, the chance you will get through a day with none of them failing is 1-(1/1000)<+>1000<+>, or about 37%. In other words, two days out of three, you're going to have to replace something.

Of course, if the mainframe fails, *all* your machines fail. However, one of the things you're paying for with your million dollars is rock-solid reliability: a System/390 is built with enough redundancy that if something fails, the rest of the system stays up and can be hot-swapped. This isn't just disks: on multiprocessor machines, you can replace a failed processor without bringing down the system. I giggled when I saw Microsoft trumpeting it had achieved 99.5% up time with NT. Thus, under exceptionally good circumstances, properly administered and maintained, NT is down, on average, a little less than an hour a week. I was a VM systems programmer from 1992 to 1994; during that time, we typically had under an hour of scheduled down time a *year*. *Unscheduled* down time was zero.

Backups cease to be an issue; because VM is managing all the disks for the virtual machines, all their data is backed up with the VM backup. The same with data integrity; for that kind of cash, you get well-implemented hot-swappable RAID, in which the complexity is never even visible to the Linux machines, because they see their disk space just as devices presented to them by VM. Basically, no matter how many virtual machines you have, you have only one actual machine to protect, so the cost of doing so remains constant, rather than scaling with the number of machines.

Furthermore, VM has an extremely efficient cache. Frequently accessed disk blocks will be held in the cache and requests never go to the drives at all. This is a huge win if you either share disks among machines, or if you're running a server farm.

There's also a low end in the mainframe market. The P/390 is a PCI card containing a chip with the S/390 instruction set on it. It is sold in combination with another PCI card which provides a channel interface (so you can drive your real S/390 peripherals), a PC running OS/2 and some driver software; you run VM, VSE, OS/390 or Linux on the card. The prices for the PC Server System/390 are under $10,000 now. There's also the slightly more expensive R/390, which sits in an AIX box. Ten thousand dollars is well within the budget of a smallish software development company. Of course, these boxes won't support a thousand simultaneous machines, but they'd do fifty fairly comfortably (they support 130 or so simultaneous CMS users). In fact, I'm planning to use just such a system to play with a virtual Beowulf cluster, among other things, and maybe experiment with MOSIX, too.

There's also a lot of middle ground between these two ends. In short, it's not much more expensive, from a purely hardware point of view, to put *N* virtual Linux boxes on a mainframe than it is to simply buy *N* boxes, and it becomes a good deal cheaper as *N* increases.

But Is It Linux?

The short answer is, yes, it's Linux. The version I'm running at penguinvm.princeton.edu is 2.2.13. The S/390 patches are already in the stock 2.2.14 source, and IBM's team is hard at work integrating L/390 into 2.3.*x*. By the time you read this, it should all have been merged fairly seamlessly into the mainline development kernel tree.

Linux on the System/390 is just as much Linux as Linux/PPC, Linux/SPARC, Linux/m68k or Linux/Alpha. The System/390 port is integrated into the main kernel source tree; unlike ELKS or RT-Linux, it's the stock kernel, rather than a subset or an extension. From a user's perspective, nothing differs between running it on a 390 and running it on an x86 machine. The shell works the same way, X applications work the same way, GNOME has been ported (by adding a couple of lines to the configure scripts) and it all just works. There's not much exciting to say, because if it comes with source, the odds are very good you can build and run it with minimal effort. As “Think Blue Linux” gets off the ground, you should be able to install a binary with RPM. At the time of this writing, they had about 420 RPMs available; there should soon be many more.

From an application programmer's perspective, it's about the same. There are a couple of very minor patches for config.sub and config.guess to get **configure** to recognize the S/390 architecture (these have already been submitted to Ben Elliston at Red Hat, who maintains **autoconf**). With its next release, configure should support L/390 with no modification. Once the patches have been applied, anything which uses autoconf builds and runs right out of the box. This

includes Bochs (an x86 emulator); apparently, some intrepid soul has built Bochs and booted the NT Server, very slowly, under L/390.

The other major issue is endianness. Unlike the x86, S/390 is big-endian. Programs which assume Intel byte order will fail. But if these programs have already been ported to, say, Linux/PPC (another 32-bit, big-endian platform), then those bugs have been crushed. This is not a 390-specific problem as such, but a general portability issue.

At the time of this writing, **pthreads** support was still buggy in the version of L/390 I was using, which caused odd bugs in VNC and the Hercules emulator; this has apparently already been fixed, but I haven't rebuilt the system with the latest patches. Likewise, there was an optimizer bug in GCC, causing it to generate bad code with **-O2** and above; this too has already been fixed, and should no longer be an issue by the time this appears in print.

Deep within the guts of the kernel, things get fairly hairy. IBM has released its kernel code and glibc modifications under the GPL, and they're available from the IBM DeveloperWorks page; it's all there and well-commented. It's fascinating stuff, especially the huge differences between x86 interrupt design and the way S/390 talks to its peripherals, but far beyond the scope of this article.

Networking is handled via either the OSA-2 (Ethernet or token-ring) device driver or point-to-point high-speed channels. (OSA is the ironically named "open systems adapter": the device driver is OCO—object code only, meaning no source.) If you're running under VM, you can set up something called IUCV (Inter User Communications Vehicle), which basically lets you establish a PPP connection to another virtual machine. If, by the way, you're beginning to get the feeling IBM has its own language, all acronyms, you're absolutely right. What this IUCV means is that it's purely internal—there is no real-world device corresponding to an IUCV adapter. You can also set up a virtual CTCA interface (channel-to-channel adapter). Channels *are* real-world interfaces, implemented these days over fiber optic cables. IUCV runs at 500MBps, CTCA at 250MBps. Communication between your virtual machines should not be a bottleneck. Writing to IBM urging them to change their OSA-2 OCO policy is unlikely to help, but might make you feel better.

Currently, penguinvm.princeton.edu uses CTCA to talk to the VM hypervisor's TCP/IP stack, but it would also work to set up a Linux virtual machine and use it as a firewall, doing packet filtering and network address translation for the machines behind it. That machine can then talk directly to either the outside world with its OSA-2 interface or VM's stack with a different IUCV/CTCA interface.

Installation

Installation is still ugly. Basically, you're not going to get anywhere unless you understand how to install a new operating system on a System/390, *and* you understand how to install and set up Linux. It's a four-step process:

- Build a boot loader image from the supplied files.
- Boot the loader.
- Dump the file system tar archive onto newly created EXT2 partitions.
- Reboot and answer the configuration questions.

The first two steps require S/390 knowledge, the last two Linux knowledge. You *may* be able to puzzle it out from the documentation, but if you're shaky in one area or the other, it's probably a good idea to find someone who has the other base covered.

Marist College has been at the forefront of Linux/390 development. Marist's page contains the tape and disk images necessary to get started, as well as documentation, and is the place to begin. There is also an extremely active mailing list hosted at Marist, Linux-VM, which is not VM-specific, but is devoted to Linux on the mainframe, with or without VM. The file system layout of the Marist disk is a little weird if you're used to Red Hat or SuSE. This will change as actual distributions get ported.

There *is* already a distribution for the S/390: “Think Blue Linux” or “Iron Penguin”, based on Red Hat 6.1. At the time of this writing, it is only a downloadable collection of packages and not yet available on CD or tape.

As soon as I get the testbed P/390 I have available in Virginia fired up and networked (which should also have happened long ago by the time you read this), Richard Higson and I will begin the process of porting Debian Potato. I'm expecting it to be fairly easy, since Debian already supports quite a few platforms.

In short, by the time this is published, installation should be much simpler than it currently is. It's not insanely difficult even now, but it is comparable to Linux/x86 in early 1993, when SLS was beginning to get off the ground. It certainly is no Lizard or YAST2, and installation requires you to understand both the System/390 and Linux.

That's Neat, but I Don't Have a Mainframe

Perhaps I've convinced you Linux/390 is cool, and you'd like to see it for yourself. Now, you probably have a problem: you do not have access to a

System/390. There are at least three ways you can play with Linux/390 without already having an actual System/390.

The least interesting option is to do what the IBM guys did initially and download their glibc, gcc and kernel patches, and build L/390 software in a cross-compilation environment hosted under Linux/x86. Sure, this works, but it's not very much fun. For the experience, you need to be running L/390 itself, which requires a System/390.

The expensive option is to buy a mainframe. A minimal P/390 on the used market will set you back less than $10,000. That's not all that much, is it? Okay, so $10,000 might be a little steep for a neat toy.

You'd still like to play, but can't spend ten grand? How does “free” sound, then? Free, as in beer *and* speech. Roger Bowler has written an intensely cool emulator, Hercules, which runs under Linux and emulates either a System/370 (the previous generation of IBM mainframe) or a System/390; it also emulates many of the more common mainframe peripherals. It's open source (not GPL, but the license is quite reasonable, basically just forbidding commercial use) and very easy to build. The emulator is sufficiently thorough to boot OS/MFT (a simpler IBM mainframe OS, circa 1966) and use MFT to build and boot OS/MVT (MVS's progenitor). Work is being done to bring MVS 3.8 up on it.

More to the point, Hercules emulates a System/390 well enough to boot Linux/390. I've put up a page explaining how; see the Resources section. Currently, it's not yet useful; there is no network device emulation, so getting stuff into and out of the machine is difficult. It can, we think, be done with the supplied tools and tape image files. By the time you read this, someone will probably have figured out how to load the Marist file system onto a 3330 image, which would allow for actual development. Hopefully, it will not take too long to develop enough network support to allow the virtual S/390 to appear on the network, in the same way VMWare machines do.

Be warned: you'll need patience. On my PII-300, Hercules takes close to an hour to boot Linux, although once it's up and running, interactive performance is not actually bad at all (I started playing with Linux on a 4MB 386/25, and Hercules is no more painful than it was). Many people on the Hercules mailing list (sign up from the Hercules home page) are aggressively working on performance-tweaking Hercules, so its speed should increase significantly. The referenced page will contain updates as we turn Hercules into a reasonable development environment for L/390, so check back often.

It's a grand adventure; we're exploring new territory every day. We need your help. Hop aboard, and bring your penguins. The dinosaur doesn't bite. Honest.

Resources

Glossary

email: adam@io.com

Adam J. Thornton has been using Linux since 0.09, making him older than he cares to contemplate. He distinctly remembers thinking SLS was for sissies. When he's not hunched troglodytically in front of his monitor, he enjoys playing with his Greater Swiss Mountain Dog Vinnie, bicycling with his fiancee Amy, and drinking Scotch, but not all at the same time. Write him at adam@io.com.

Archive Index Issue Table of Contents

Advanced search

Game Developers Conference 2000

Jason Kroll

Issue #74, June 2000

Yes, I have been to the Game Developers Conference 2000 in San Jose, and I have such news as has not been reported in this column for a long time.

A video game convention, such a thing exists; and to such a place I went, and back from such a place I am to yield a report to such readers as subscribe to a computer magazine about a kernel and its operating system. Yes, I have been to the Game Developers Conference 2000 in San Jose, and I have such news as has not been reported in this column for a long time. In fact, since I don't usually cover commercial software, such news has never been reported here. Alas, variety is more than just the spice of life—it's also a reproductive advantage, so if we want to reproduce computer games, we'll have to learn how to crack them. Actually, that's one of the many things that has changed since I last visited commercial games. It seems the sheer size of the software is the best copy-restriction scheme of the day. No longer have we any clever challenges to break (and let's just say it was more fun cracking a game to look at its code and make it do funny things than it was to be able to copy it. As a kid, I used to rewrite dungeon adventure games in skater lingo). Many things have changed, and it is with a heart not quite as heavy as a heavy heart, yet not quite light as a light heart, that I report the findings of my undercover operation.

First of all, video game development is a serious industry. It's like the movie industry, except it isn't suing everyone on the Net. There's a struggle for the teenage mind and dollar, and the gaming racket is not so much about creativity as it once was, it's not about a single developer sitting in his bedroom churning out the latest big hit. Producing a video game is probably more difficult than producing a film, and there's a whole infrastructure built up to support the gaming industry, which brings me to my next point.

Video game development requires infrastructure, not just the infrastructure within the computers themselves that lets game developers have easy and uniform access to the video display, sound card and I/O devices (an infrastructure which we need to develop), but an actual support system within the industry to supply graphics engines, 3-D rendering tools and development kits. You may have noticed that modern games tend to be 3-D remakes of the Quake variety. Well, in order to produce the graphics for these games, developers have to use special graphics packages to develop 3-D models. Then, they quite often buy someone else's graphics engine to use. Some people at the convention were even trying to sell audio samples.

Another thing to know about the gaming industry is that it doesn't pay like a job at Microsoft. The top few games take up nearly the whole market, and everyone else is a tad out of luck. Now, creative people like Sid Meier and Richard Garriot who consistently make enormous hits aren't starving in the streets, but they aren't as wealthy as Bill, Paul and Steve, either. Why is gaming industry pay important? Because they're recruiting.

Yes, we coders are in demand, at least right now, and the video game industry has to be one of the most interesting areas to work in. At the moment, I don't do the “proprietary thing”, so it's mostly off limits to me unless I change my mind someday. But, if you want a fun place to work where you can exercise creativity, be aware the video gaming industry is hiring (pre-IPO with stock options, even).

Computer games can seem stupid, especially the new rash of boring 3-D killers, but they've vital to the success of computers. They attract future hackers, this is true, but their real value is that they drive the hardware industry. Of course, Microsoft's bloated OS also forces people to keep buying bigger and faster machines, but the main driver of progress (or so I choose to believe) is video games. Graphics cards, sound cards, processors—the more people buy games, the more these technologies are driven; in the end, we win.

What's new on the proprietary front? Well, if you haven't played modern games in a while, you might be impressed by the state of current production quality. However, if you have, you might be bored with all the bloody 3-D shooters. I didn't see any games that demonstrated truly new ideas, but at least the developers are getting better at implementing the old ones.

For GNU/Linux, I can say that Loki is our best hope right now. With seven releases so far and *twenty* scheduled this year, don't go back to dual-booting; there are more games on the way, and they're going to sound better, too. Loki is spearheading the OpenAL project, which is a cross-platform 3D audio library, essentially the OpenGL of audio. It's LGPL (lesser GPL, as it is now called), meaning that even though it is guaranteed to be free, you can use it to make things that aren't. It's a pragmatic approach, if we want it to be accepted in the hyper-commercial industry. Check out http://www.openal.org/ if you're interested in this project.

One thing holding back video game development on Linux is the lack of internal infrastructure. The multi-tasking, multi-user nature of our OS presents a slew of trouble when one user wants to play a game that needs access to the frame buffer, audio device and I/O ports, as well as some assurance of soft real-time performance. How do we avoid resource conflicts? How do we give programmers a consistent way to talk more-or-less directly to the frame buffer or the digital signal processor? And, how do we solve the latency problems? Fortunately, these questions are being forcibly answered by the imposition of demanding games. Once the infrastructure is there, we can expect easier development of both free and proprietary wares for GNU/Linux.

Didier Malenfant brought another ray of hope. Already quite a successful game programmer (with a legacy from the Amiga demo scene), he gave a presentation for developers interested in developing games that are easily portable to Linux. As long as developers start with several platforms in mind, they won't encounter much trouble porting later on. I spent much of my time at the show trying to explain the details to numerous intrigued developers, so I have a feeling we've planted the memes pretty successfully. In fact, the ideal of many developers is to be able to work under Linux and then cross compile to other platforms. Jon Taylor of Crack.com considered Linux the ultimate development platform, a sentiment I'm sure nearly all of us share.

Hardware news? Well, as you know, the Amiga is coming back again, this time hopefully for real. It is rumored to be on a Linux base, but then we've been living on rumors long enough to build a whole new operating system. Microsoft announced their XBox, and several Microsofties got quite haughty with me when I offered them a free *Linux Journal*, so I perceive less than good karma in the air surrounding that outfit.

As for the future of video gaming, some people expected computer gaming to be completely replaced by consoles, others thought the two would coexist without trouble, and still many thought that console and computers would offer different kinds of games. Personally, I can't understand the value of a gaming box if you can't hack it, so I hope they don't get too popular.

Surprise, surprise—more things work on Linux than we know. Of course, I've been asked by certain firms to keep silent about what exactly it is that works, but let's just say the industry has noticed our little OS project, and a lot of software is either being ported or has been ported and is now sitting around going stale. This is one more reason I suspect if we get our audio and video acts together, the gaming industry on Linux will be ready to take off.

Anything special to watch out for? Well, I saw some demos from Aegis Simulation Technologies (creators of the graphically impressive BFRIS), one of which will come out soon and looked neat, and another which is being delayed and looks fantastic. The projects are still under wraps, but keep on the lookout: if you're into gaming and don't mind the proprietary thing, you probably won't want to miss these. Now, enough with the proprietary bit and on to free software, since that's what we're all about, no matter how often we may forget.

Bertrand Meyer, in his criticism of the ethics of free software, said that commercial software gave a “proof of concept” that free software then copied. So my question would be, what can free game developers learn from the commercial video gaming industry? Well, we need an infrastructure, both externally in terms of ultra-powerful libraries, rendering tools and development kits, and internally by resolving once and for all how we can reconcile real-time multimedia with a multi-tasking, multi-user OS. I imagine there are a million more-informed opinions than mine floating around, but I shall meander nonetheless! Nothing beats **gcc** for a development tool, and the OpenGL/MesaGL libraries seem quite easy to use. As for rendering tools, hopefully Blue Moon, renderman, GNU's Panorama and PMR will keep progressing. It seems as though virtual reality and artificial intelligence, those trendy business ventures of the 80s which left craters all over Silicon Valley, are finally coming back, so for VR we'll really depend on 3-D engines and rendering tools. As for AI, I wonder if NPCs (non-player characters) can be written in LISP, and does this mean I can use it again? Video game development is so interesting; it would

be nice if we were able to develop for free, share our libraries, and not have to reinvent the wheel with massive production teams every time around. I wonder if we will see the day when a single programmer, or a handful of close friends (Cathedral alert!), working only with software released under the GPL, could spit out a truly creative masterpiece on a par with the best of the commercial offerings. One would hope.

Linux Journal's technical editor **Jason Kroll** (info@linuxjournal.com) suspects the ability to create a world in our own image is a more powerful incentive in the development of GNU/Linux than the supposed value of peer status.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #74, June 2000

neomail, cscmail, Twig and more.

Games, as I mentioned in a previous article, are the unquestioned staple that have brought the masses to computing. More man years have been wasted playing solitaire or other games over the last decade than on most other non-productive pursuits. Obviously, this is not why computers are so popular with business. Originally, spreadsheets were the reason for computers. Later, databases also provided a lot of the impetus for buying systems (at least the larger systems; smaller systems were used primarily for word processing). Today, with most companies enjoying Internet connections, most employees of almost any size business will tell you they use their systems mostly for e-mail. From senior management to secretaries, e-mail has redefined how companies do business. This also applies to individuals. You can send just about anything via e-mail, including money. So I'll devote half of this article to e-mail. In fact, I could devote several articles to e-mail, so I'll restrict this to some of the better new MUAs (Mail User Agent) I've found.

neomail: http://www.sourceforge.net/neomail/

With **neomail**, you can read your mail in a web browser (secure server) if you want. It has support for an address list, and you can specify how large (in KB) the list can be to keep users from getting carried away. You can import a comma-separated value (.csv) file into the address list, which is handy if you're coming from Outlook or Netscape Mail. With this MUA, if you can connect to the Internet, you can read, compose, delete, forward, file and generally handle your mail. It requires Apache, Perl, suidperl, Perl CGI module and sendmail or exim.

cscmail: http://cyberdeck.org/cscmail/

This particular mail client claims to be a clone of Outlook Express. I've never used OE, so I don't know if it lives up to that claim, but it is a well-done mail client that includes drag and drop, easy filters for sorting, mail folders, address lists and more—all in a Perl-GTK wrapper. The most difficult part was installing the Gtk::Perl module, which requires you to build the main part and several subparts. It requires gtk, GtkXmHTML, Perl and the following Perl modules: DBI, Text::CSV_XS, SQL::Statement, DBD::CSV, DateManip, Mail::Sender, HTML::Parser, Gtk::Perl and Mail::POP3Client.

Twig: http://screwdriver.net/twig/

Twig does a very good job with mail. But Twig does more than just mail, it also attempts to be a Personal Information Manager (PIM) by providing you with a Contacts list, a Schedule, a ToDo list, Bookmarks and access to News servers. The schedule is an appointment list and provides a small calendar on the side. Within Twig, you can create groups for information sharing. This does a very nice job for an office or workgroup and includes support for various databases. It requires MySQL (or Postgres or Oracle) and Apache with PHP3 that includes support for IMAP.

AeroMail: http://the.cushman.net/reverb/aeromail/

AeroMail is a very spartan mail reader that will allow you to read, compose, file and delete messages, but very little more. No address books or contact lists are included—it's simply fast and efficient. If you don't want to give your users space for an address book, you'll like this mail client. It requires Apache with PHP3 that includes support for IMAP.

tnef: world.std.com/~damned/tnef-latest.tar.gz

Bugged by those absurd, non-standard tnef files Microsoft likes to pollute e-mail with? This little utility will allow you to read them if you're wondering what's in there. It requires glibc.

apachedb: http://www.goofy.gaudi.dhs.org/

Need something to track usage of your website(s)? **apacheDB** allows you to register all hits in a MySQL database and uses PHP3 to query the database for agents, hits, referrers, usage and more. You can convert your present access_log into the database and continue from there, if you wish. It displays hits graphically as well as numerically. You may need to make some small modifications to the script if you don't use the combined logs, but the author has some examples. Your marketdroids will love this tool, and your CFO will like

the price. It requires MySQL, Apache with PHP3 support, Perl, Perl modules DBI, DBI::DBD and Entry.

mozart: sourceforge.net/project/?group_id=2466

Mozart is a start on a PIM. While not all areas are completely functional, and some work needs to be done on aesthetics, the major areas are all in place. This PIM is designed to be used by a group and currently supports Contacts, Calendar, ToDo list and Appointments. It requires a web server with PHP support, MySQL and a web browser.

MultiNet: http://users.capitolonline.nl/~nlco5954/multinet/

Do you carry a laptop back and forth between home and work? Both sites have their own network? Both have different network setups? If you answered yes to the above questions, you might want to check out MultiNet. This little utility takes care of configuring (or reconfiguring) the network for you. It will at least save you manual configuration of each location. Can be launched from a command line or from your rc scripts at bootup. It requires /bin/sh and dialog (slated for rewrite in Perl).

wakeup: http://soul.apk.net/wakeup/

This small utility will provide you with a wakeup call at the time of your choosing, to the music of your choice. Provided, of course, you have an MP3 for it to play. It requires Perl, mpg123 and at.

email: dbandel@pananix.com

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.

Archive Index Issue Table of Contents

Advanced search

Embedded Linux News Briefs

Rick Lehrbaum

Issue #74, June 2000

The latest in embedded Linux news.

SGI and **SuSE** are collaborating on a version of IRIS FailSafe for high-availability applications in Linux environments. “High availability is an important piece of the puzzle,” says Linus Torvalds. (www.sgi.com/, www.suse.com)

Samsung launched “Yopy”, a PDA with Linux inside. The device contains an ARM processor, a four-inch color LCD, HTML browser, MP3 player and CompactFlash expansion slot. Add-ons will include a TV tuner, GPS, wireless modem and digital camera attachment. (http://www.samsung.com/)

National Semiconductor will offer Citrix Systems' Independent Computing Architecture (ICA) software for its Geode single-chip Internet appliance. The Geode chip, which runs Linux, contains a 64-bit “X86” CPU, plus PCI, video, audio and power management functions. (http://www.national.com/appinfo/solutions/)

NETsilicon announced NET+Lx Linux connectivity software for its NET+ARM one-chip embedded appliance. The chip contains a 32-bit ARM7TDMI RISC processor, plus Ethernet, SDRAM, SPI and serial interfaces. Display connection requires an additional component. The OS used is uClinux. (http://www.netsilicon.com/)

Ericsson showed off a Linux-based cordless Screen Phone (HS210) that combines wireless connectivity with Internet access, telephony and e-mail functions. (photo: www.ericsson.com/pressroom/screen02.htm)

Lineo acquired **Rt-Control Inc.**, the developers of uClinux. uClinux supports microcontrollers, including Motorola 68000 and ColdFire, ARM7, Axis ETRAX and the Intel i960. (http://www.lineo.com/)

Two new embedded Linux whitepapers are online: “Embedding Linux in a Mobile Robot” and “Using Linux in Embedded and Real-time Apps”. (www.linuxdevices.com/cgi-bin/article_view.cgi?artid=AT3782871866, www.linuxdevices.com/cgi-bin/article_view.cgi?artid=AT3611822672)

InfoCharms unveiled a tiny, brooch-style, Internet-ready communicator that contains Nanix, a slimmed-down Linux. “We hope Nanix will become the Linux of wearable computers,” say company representatives. (http://www.infocharms.com/)

“Which Linux will you use in future embedded system projects?” The answer: **Lineo Embedix** tops the embedded distributions list, followed by **PROSA ETLinux** and **Lynx BlueCat Linux**. Red Hat, Debian and Caldera were the most popular standard distributions for future embedded designs. Nearly two-thirds said they will pay for outside services and support. (http://www.linuxdevices.com/polls/)

Espial announced an embedded Internet browser that fits in under 800KB. (http://www.espial.com/)

Atmel announced a Linux-based single-chip Internet appliance, the AT75C310, which includes support for VoIP (Voice over Internet Protocol) and audio. The device contains an ARM7TDMI CPU, RAM, DSPs, SPI, USART and parallel I/O. (lel I/O. (http://www.atmel.com/)

FSMLabs released version 2.2 of RTLinux which, among other enhancements, offers increased POSIX compatibility. (http://www.fsmlabs.com/)

Axis Communications began delivering ETRAX 100, a one-chip Linux-based Internet appliance. It contains a 100 MIPS 32-bit RISC processor, as well as Ethernet, IDE, SCSI, parallel and serial ports. (http://www.developer.axis.com/)

Sigma Designs announced a “set-top box” reference design based on Intel's Celeron processor and 810 graphics controller, plus Sigma's REALmagic video decoder chip. (http://www.sigmadesigns.com/)

MoJo Designs Eyelet GUI now supports Linux. The processor-independent GUI helps developers of Linux-based embedded systems avoid the substantial overhead of X Windows. (http://www.mojodesigns.com/)

PenguinRadio announced plans for a low-cost Linux-based adapter that lets you hear thousands of Internet radio stations through your stereo system. The company will also produce a version for cars. (http://www.penguinradio.com/)

Compaq's Western Research Center is developing “Itsy”, a Linux-based open platform for pocket computing. The purpose of Itsy is to foster development of novel user interfaces, applications and research projects. Itsy uses a StrongARM CPU. (http://www.research.digital.com/wrl/itsy/)

Everybook Inc. announced a Linux-based electronic book containing **Lineo's** Embedix Linux OS and Embedix Browser. Users can purchase and download digital books and magazines from the Everybook Store. (http://www.everybook.net/)

EL/IX version 1.1, Red Hat's proposed spec for adding real-time features to Linux, is now available for download. (http://sourceware.cygnus.com/elix/elix-api.pdf/)

MontaVista founder Jim Ready discusses his views on embedded and real-time Linux and his company's plans and strategies in an interview. (www.linuxdevices.com/cgi-bin/article_view.cgi?artid=AT9093728800)

Victor Yodaiken has received US patent 5995745 for a technique of running a general-purpose operating system (e.g., Linux) under a multi-tasking kernel (e.g., RTLinux). (www.patents.ibm.com/details?&pn=US05995745__)

International Data Corporation (IDC) says the “Post-PC Era” has arrived. Their new report projects that shipments of consumer information appliances will outpace those of consumer PCs by 2002. (http://www.idc.com/)

IBM announced ViaVoice, a set of speech recognition technologies and tools for Linux, saying this is the first in a series of forthcoming IBM Linux solutions. (http://www.ibm.com/software/speech/)

Kerbango, Inc. announced the world's first standalone Internet radio. The Linux-based device lets you access web-based streaming audio without a PC. The OS inside is MontaVista's Hard Hat Linux. (http://www.kerbango.com/)

Will **Microwindows** be the GUI of choice for tomorrow's Linux-based embedded systems and appliances? The open-source GUI offers both a “Win32 GDI API” and a minimized X-compatible GUI. (http://www.microwindows.censoft.com/)

ISDCorp has released Royal Linux for systems that use non-Intel CPUs such as ARM, MIPS, Motorola 68K/Coldfire and PowerPC processors. (http://www.isdcorp.com/)

MontaVista Software will use RTLinux, from FSMLabs, as an “accelerator” option for embedded market customers wanting “hard real-time” features. (www.mvista.com, www.fsmlabs.com)

Lineo has acquired real-time Linux specialist, **Zentropix**. (www.lineo.com/, www.zentropix.com)

Lineo's CEO Bryan Sparks discusses his company's strategy of charging royalties for Embedix Linux in an interview. (www.linuxdevices.com/cgi-bin/article_view.cgi?artid=AT9593643351)

Eagle Wireless International unveiled that it has received a contract worth $125M US for 500,000 Linux-based set-top boxes. (http://www.eglw.com/english/)

email: rick@linuxdevices.com

Rick Lehrbaum may be reached at rick@linuxdevices.com.

Archive Index Issue Table of Contents

Advanced search

Location Times Three

Stan Kelly-Bootle

Issue #74, June 2000

Stan on books.

Over the years, I've maintained a list I call “Misshelved Books—the Librarian's Nightmare”. Thus, *The Book of J* is found among Iverson's Programming Languages and *The Bourne Conspiracy* in the UNIX Shell section. Search the tiny Linux Driver shelf, and you'll probably find P. D. James's *Devices and Desires*.

And where else but in the bulging OO (Object Orienteering) category would you expect Alan Bennett's *A Private Function*, Marlo Morgan's *Mutant Message Down Under* (surely a Dinkumware production) and Baudrillard's *le systeçme des objets*?

The latter, one of those impenetrable culture-theoretic works the French are so good at, probably merits its place alongside the growing number of mystical and pseudo-philosophical books on OO as the one true path to code nirvana. But to avoid memory leaks, remember that each constructor must be followed by a deconstructor.

A more general, cynical observation notes the large number of computer manuals filed under Non-Fiction.

Springer-Verlag's math lists have some strange titles just begging for misclassification: *Equimultiplicity and Blowing Up*, by Herrmann, Ikeda and Orbanz, surely belongs in the Balkan History section—and what about *Motions of Coupled Bodies about a Fixed Point*? Don't ask. Yet, there was that great, deliberately teasing book, *The Joy of X*.

Now dated is the joke that the *Norton Classics* series escaped misfiling under DOS 8080 Assembly Coding because the great man's mugshot did not grace the

covers. Also fading from our collective memories are tales of *Five Graves to Cairo* and *The Chicago Style Guide* being catalogued under Microsoft Vaporware.

I've also noted some non-technical LibCat glitches: *The Trials of Oscar Wilde* under Humor, Anai<\#239>s Nin's diaries under Fiction (well, one never knows) and *Black Beauty* under African-American Studies.

The quirks of ASCII sorting by title show up in weird librarian contiguities. Thus, immediately after *Software Engineering with Ada*, one booklist offers Rich Zubaty's *Surviving the Feminization of America: How to Keep Women from Ruining Your Life*. If Lady Lovelace were alive today, she'd be turning in her grave!

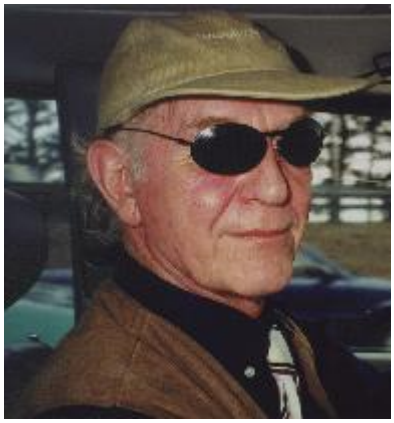
We have happily left behind us those tab-card restrictions whereby titles were crudely curtailed to fit 80 (and even 22) columns. I still treasure invoices for "WHYIAM" (Russell's *Why I Am Not a Christian*) and (shades of Edward Lear) "MOZDONG" (Mozart's *Don Giovanni*). As Fritz Spiegl, my favorite musicologist, remarked, "From the sublime to the cor-blimey."

On the bright side (there's always a bright side, except perhaps for Java Standards), misplaced books add to the joy of browsing. Recall *The Three Princes of Serendip*, the heroes of which had the fairy-tale gift of accidental discovery? Horace Walpole post-coined the word "serendipity" in 1754, and with its own touch of magical self-reference, the word survives. You can find it in action each time you visit your public library's Recent Additions section, a random set of books *awaiting* misallocation.

Which is where I first spotted Eric S. Raymond's *The Cathedral & The Bazaar* (O'Reilly, Sebastopol, CA) in the hard-covered flesh, as it were. On my next visit, the book had moved to—who knows where? Medieval Economics? Church History? The clever numerical system (which I've dubbed Dewey-It-Yourself) presupposes a categorical knowledge of content in areas where the categories are forever shifting. Eric's book, following expansive explanatory titular trends, does have the helpful subtitle *Musings on Linux and Open Source by an Accidental Revolutionary*, so I hope it found its way to the "appropriate" shelf. Or rather, with duplicate copies, to several shelves, since *The Cathedral & The Bazaar* is aimed at a wide audience, including those who do not usually visit the Computer section.

In the meantime, I've acquired my own copy (via amazon.com [shameless plug]) and plan an in-depth review in next month's column. Rather late, I fear, since everyone from Linus Torvalds and Tom Peters to Guy Kawasaki, from Brian

Behlendorf and Wayne Caccamo to Larry Augustin (and more) has stamped their approval. Yet, I will try to be objective—not easy, since Eric and I have collaborated on and off over the years on hackers' lexicography, and he has graciously reviewed one of *my* books.



Stan Kelly-Bootle (skb@crl.com) has been computing on and off since his EDSAC I (Cambridge University, UK) days in the 1950s. He has commented on the unchanging DP scene in many columns (“More than the effin' Parthenon”--Meilir Page-Jones) and books, including *The Computer Contradictionary* (MIT Press) and *UNIX Complete* (Sybex).

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #74, June 2000

Readers sound off.

Send A Message to Laurie

Laurie Dare's letter on the Linux Open Source Expo and Conference Program in Sydney (Letters, April 2000) struck a chord with me, and I decided to answer the letter.

First of all, I will mention that this was my third trip to Australia in the past twelve months, where I have given talks at much-lower-priced venues, such as CALU and AUUG, as well as free talks to various user groups in Australia.

The $1350 AUS price quoted by Laurie in the letter was the price for both the Linux conference and the ASPCON conference. The Linux conference by itself was $695 AUS, with a Government/Academic discount of 10% *or* an active LUG member being able to get in for $595 AUS. The dinner price was correct as stated at $125, but again I would like to point out for our non-Australian readers that this was $125 AUS and not 125 USD. Since USD were trading for 1.82 AUSD that day, it means the dinner really cost about 68 USD, not an unreasonable price for a dinner in Sydney.

Nevertheless, I was a bit upset to hear the dinner price was even that high, and I spoke to the organizers about it. They assured me that the dinner was a “break-even” event. I can imagine that, since the Sydney Linux User's Group tried to rent a room for a four-hour meeting in the same facility, and were charged $500 AUS (274 USD) for the privilege of having a room and chairs. This $500 AUS charge was picked up by LinuxCare, much to the relief of the SLUG.

Still not completely mollified by the explanation, I proceeded to put on five more one-hour talks over the next three days of the exhibition, at the small theaters in the Red Hat, LinuxCare and SuSE booths. These talks were free of

charge, and were well received by the people who attended. Other "luminaries" such as Bob Young, Dr. Andrew Tridgell, Paul "Rusty" Russell and others talked in these theaters as well.. . . Warmest regards, —Jon "maddog" Hall, Executive Director, Linux International, maddog@valinux.com

Just an Old-Fashioned Guy

I'm happy to find a page ("Take Command", March *LJ*) on the old-fashioned stuff. I love old-fashioned stuff; it saves a lot of time and typing. However, your article doesn't do justice to **uuencode**/**uudecode**. There is no need at all to save stuff on disk twice. I often just type:

```
tar cvzf -
  mail -s subject recpt@domain.com
```

which results in less typing, less waiting, less stuff to be deleted.

And the recipient doesn't need to save the mail and then **uudecode *file***. It's easier to use the | (pipe) command of the mailer: **|uudecode** and the (compressed) file is output directly on disk.

—Alessandro Rubini rubini@gnu.org

Well, you are right, of course. However, the "Take Command" column is considered a newbie column, so I like to keep it short and simple. I didn't want to get into explanations of pipes or mailer options, both of which have been discussed in other articles. With **mutt** and other mailers, you don't even have to do the uuencode any more; just choose the attach option and it is MIME-encoded, then sent off. I was called to task by someone else for not mentioning that one —Editor

Can You Spell Linux?

"Office Wars: Applixware and StarOffice" (Jason Kroll, February 2000) was an interesting and informative article. I discovered an enlightening tidbit while using the StarOffice suite. We can only hope that Sun's commitment to the Linux community is not reflected in the fact that typing "Linux" into the StarWriter application produces a spelling error.

—Domenic R. Merenda dmerenda@geeks404.com

Whooping It Up with Apache

I was reading through issue 71 of *Linux Journal* when I happened across Raju Mathur's excellent little article on using the Apache proxy server to suppress

banner ads. The technique is very interesting and fun to play with. I did, however, have a problem.

Setting up Apache as a proxy was a breeze. Getting **mod_rewrite** to work was not. It was only after some perusing of the Apache mod_rewrite documentation that I found my problem. Whether you are compiling Apache with mod_proxy and mod_rewrite, or loading them as modules, you *must* load mod_rewrite after mod_proxy or it will not work.

On a side note, I think it should at least be mentioned that many web sites make money (or at least try to alleviate their costs) by running the banner ads. Anyone who wishes to set this up should at least give that a thought before actually doing it.

—Stephen Carpenter sjc@delphi.com

Slink Not Debian

I just finished reading your review of "Learning Debian GNU/Linux" at http://www2.linuxjournal.com/lj-issues/issue71/3821.html .

Having owned this book and spent a good deal of time conversing with both O'Reilly and VA Linux, I think you have made an error that deserves mentioning. You state near the end of your review, "A CD-ROM containing the Debian 2.1 distribution is bound into the book." This is incorrect.

What is actually bound into the book is a copy of VA Linux's "slink and a half" modified version of the Debian GNU/Linux 2.1 distribution. This is *not* the Debian GNU/Linux 2.1 distribution, but a Frankenstein hybrid of the stable 2.1 (slink) distribution and the unstable 2.2 (potato) distribution. The new user may have some difficulty getting help with this hybrid, and VA Linux will not provide any support except to state that it is *not* Debian 2.1, and if you want support, you should purchase their boxed distribution. Please inform your readership of this state of affairs.

Thank you for your time in this matter.

—Richard Coleman richard_coleman@jws.com

Yet Another Linux University

I very much enjoyed the March issue of *LJ*. It was one of the best I had seen in months.

Since the focus of the issue was training, I would like to bring to your attention a project you may not be aware of: Linux University (http://www.linuxuniversity.org/). This is not the SGI training program of the same name, but a community-based effort to provide free Linux training to the free software community.

LU has been in existence for about two years, but has begun gaining momentum only in the last six months or so. The current course offerings are geared primarily toward developers, with classes in C and Perl, both of which have been taught in Nashville by members of NLUG. However, plans are in the works to offer classes entitled GTK+, Using the GIMP, Inside the Kernel, Linux Security Primer, Introduction to TCP/IP and Linux System Administration to the Web and Finding a User Group Near You.

The distinction of the project is that all training, whether it occurs in cyberspace or meatspace, is free of charge (although donations are welcome), classes are taught by members of the Linux community, content must be released under an open-source license, and it actively seeks to involve local Linux user groups.

If any of your readers would like to help, we encourage them to drop by the Linux University web site, check out what we have so far, and get their hands dirty filling in the holes. Thanks!

—Rob Huffstedtler rob@bagpipe.com

Mo or Les

Glen Wiley's Motif/Lesstif Application Development article in the August 1999 *Linux Journal* was a real standout, especially with the code downloadable from your web site. The article was packed with directly usable, accurate information.

The code was a model for any programmer who wants to set a high standard. The style and format was extremely readable. The comments struck a good balance between enough information and avoidance of excess.

I used this article as a guidebook to bring my limited Visual C++ and Visual Basic experience into my Linux proficiency. In two weeks, I was able to work through the program and get a respectable foothold in graphics programming in the most established and robust graphics platform in the UNIX/Linux world.

It is a very good guidebook that can accomplish that.

—Robert G. Young rgeyoung@a-znet.com

Better Late than Never

I am sorry this is late, however ....

I just want to tell you that I think your short article (“Red Hat buys Cygnus”) in the January 2000 issue is on the mark. I too share some of the same concerns.

—amgursa@cris.com

Painting the Desktop

I was reading Michael Hammel's “Artist's Guide to the Desktop, Part 2” (and enjoyed it very much, by the way) and noticed the reference to GKrellM in the section on epplets. I appreciate his mentioning it, but would like to let you know that GKrellM is no more an epplet than is **xmms** or any other Gtk application.

Perhaps the source of his impression is that initially it became most widely used by Enlightenment users, but there are actually many KDE, AfterStep, Blackbox, etc. users. While I realize Mr. Hammel would probably not have mentioned GKrellM at all in your article had you understood this, I still want to nudge the record back to what my program actually is.

I have subscribed to *Linux Journal* for several years and have always found your columns very worthwhile. I hope you have many more articles to come.

—Bill Wilson, bill@gkrellm.net

Most probably. I enjoy working with them and have many ideas. Thanks for the kind words!

—Michael J. Hammel mjhammel@graphics-muse.org

Doc is Rocking the Boat

Mr. Searls, I've just read your article “Now what ...” in the April 2000 issue of *Linux Journal*.

My relations with salespeople have always been on the cool side. It has not mattered much whether the guy is trying to sell me something or if he is a colleague. And if sales guys get the fridge treatment, then marketing guys have been worthy of only the freezer technique.

I'd like to lodge a complaint. You are rocking my boat. You are moving my perspective. You are changing my world! I've been reading *LJ* for a couple of years and eyed this “marketing guy” who came onboard about a year ago with

fear, uncertainty and doubt. Gradually that has changed, and I'm beginning to realize, “Hey! Maybe some of these marketing guys aren't so bad after all.”

I'm attending part of a Linux seminar next week. My boss okayed it immediately (when I reminded him of my request, that is)--a year ago, he would not even have considered it. It's not only the world of computers that is changing; it is *the* world. And, yes, we need marketing to achieve world domination, but maybe we can change the rules there as well. Articles like yours are changing the rules, attitudes and preconceptions.

Doc, you're all right.

—Stein Bjorndal stein@proact.no

Archive Index Issue Table of Contents

Advanced search

More Letters

Re: Raritan Article

I was absolutely thrilled with having my article published in the May, 2000 issue of *Linux Journal*.

However, and not to detract from the wonderful job you all do putting the magazine together every month, you got my email address in the bio absolutely wrong. Not only have I NEVER had the email address alex@concentric.net, I haven't had a concentric.net email address for over one year. In addition, as you can see below, my email address does not contain the name alex, nor is it on the concentric domain. If you could make a printed correction as soon as possible, and one on the Website, if needed, I would greatly appreciate it.

Thank you.

Warmest regards,

Alex Heizer, linuxaddict@crosswinds.net

"The [not so] Wonderful World of High-Speed Internet Access"

In your issue of April 2000, Jason M. Schumaker hit upon the very same issues I had when I first entered (as his article is entitled) "The [not so] Wonderful World of High-Speed Internet Access." I too, thought that having fewer companies to deal with would mean less hassle and finger pointing. Over half a year later of DSL I can faithfully relate that while this assumption has been proven correct (in my situation) a much larger issue has been revealed. Although providing the best telephone service in the world, U.S. telcos seem to be delivering third world DSL service. Phone companies seem to have no incentive to provide dependable DSL with the apparent belief that this new market is a birth right.

I've set up my own web server (DSL is definitely the way to go here) for an audio streaming site (www.the-cosmic-forces.net) and therefore desire uninterrupted service. I signed up for 1 year with my regional telephone company as my ISP and DSL provider. Without fail, service drops out for (at least) a few minutes every few days and a few hours every week. Outages usually occur after 1am when their late night shift has gone home suggesting how jury-rigged their access to the cloud must be. (They don't seem to know what redundancy is.) Unfortunately, this is the busiest time for my overseas visits.

And almost every weekend, late at night, my city goes down. (I call up friends to verify.)

More alarming however is that my email password was hacked by a high school student who entered this company's entire DSL network. Amusingly, whenever I attempt to change my password, the company's web site refuses to recognize my Linux machine (so fixes must be done on a Mac or Windows). (My “free” home page has the same problem.) Thankfully, this hacker (and his friends) have turned out to be good sports and (as far as I know) did no harm.

One huge benefit of being the only UNIX based system with my local switching center is that when I do have service, it is unbelievably fast. (Yes, I know that it shouldn't make any difference but you haven't seen my downstream/upstream speeds.)

I am planning on setting up a mirror site (again Linux but running the new open source version of Quicktime) in a different city but which is still within the same telco region. This time I'll be trying out some of the smaller, specialized DSL service companies to see if there is any difference. Yes, Mr. Schumaker is correct, they are more expensive but maybe they will deliver.

—David Koenigsberg , David_Koenigsberg@the-cosmic-forces.net

---

magazine

Background: I am new to Linux, just trying to begin learning. I have RedHat Gnome on a 466 machine.

In your magazine, the Journal, it would help my 86 year old eyes if you would color the background for articles in a little lighter shade. The use of colored backgrounds makes it difficult to read the text. I don't understand a lot of what I read there anyway, but someday I will.

Enjoy thumbing thru the magazine anyhow. Regards.

—Henry B. Poole, hpoole@scoot.netis.com

Ranting and raving (in case of unreadability, please contact

I read with some interest the letter by Scott Moore in the April 2000 issue. The way this person talks reminds me of discussions I followed some years ago on comp.os.linux.advocacy, and strangely enough, it is always the M$ advocates that get angry. I get the feeling that somewhere deep inside of them, they doubt their choice of M$ software. I have read the reply of Linus Torvalds about using M$ software, but I feel that I have to express my point of view against M$. Very short : I DO NOT LIKE MICROSOFT AT ALL.

I do not hate them, but I do not like them. This point of view was developed in less than a few years time, beginning in 1990/1991, when I discovered that IBM, Compaq and Digital Research had better DOS systems than M$ (remember the infamous DOS 4.0 ?).

I also had a few tries at Windows 2.0 and 3.0, but I had already worked with Apple Mac (I installed them) and found the M$ thing so severely lacking that I decided to stick with DOS.

One of the things that I did in years that followed, was create systems which did not have any software from M$ running. I used DR-DOS, Lotus 1-2-3, WordPerfect Office, FoxPro (before its acquisition by M$) and Zortech C++. Linux nowadays makes it even easier to run computer systems without any M$ code.

That brings me to my next point, an initiative I would like to call Zero M$, or ZM$ for short. This initiative would consist of two layers, the first being the M$ Free System (or M$FS). To create this layer, one should certify that the system being added to the layer is completely free of M$ software and then label it with a sticker. The second layer is the M$ Free Zone (or M$FZ). A zone can only achieve this status when all systems in the zone are labeled M$FS. A final status could perhaps be added, Certified Zero M$ (CZM$), for a complete company or division.

The only thing I lack are two logos which at first sight describe to someone the status of the system or the zone. Anyone interested ? Am I serious ? Who in the Free Software/Open Software is ever completely serious ?

As an aside, I try to mention nowhere the name of the Particular Company, but use M$. Aside from being used in many places, am I right that is perhaps impossible to register such a word as a trademark?

—Jurgen Defurne, defurnj@glo.be

Reply to "ranting and raving"

Hello, I am writing this letter in response to Scott Moore's "ranting and raving" letter in the Apr 2000 edition of *LJ* Scott: I almost thought your letter was an April Fools prank, and I still hope it is to be honest.

You, Scott; have written possibly the most ill-informed letter pertaining to Linux ever. Gee whiz brainiac, did you ever consider going to an FTP site and finding the Linux downloads there? Or are you such a complete simp that you have to have a nice lil' hyperlink to get you there? As far as there not being software for Linux; again, Einstein, you have to actually look for it, I have midi sequencers, graphics editors of all shapes and sizes, and anything you could imagine, even an Aol Instant Messenger clone for Linux.

As far as Microsoft "earning" their market share, (HA HA HA)the whole reason they have such an overwhelming percentage of the market in the first place is because of the fact that most average PC users are unwilling to be challeneged in ay way shape or form and continue (much like yourself) to live in ignorance of alternative OS's.

In conclusion, all I can say is you sound like you have no clue whatsoever of what Linux is about, thus perhaps you should shut the hell up until you actually learn something about the software your'e criticizing.

P.S. “Microsoft has created some of the best software on earth”

I didn't stop laughing for ten minutes.

—Zachariah Zinn, solvac@hotmail.com

Help with EFM

I have heard a lot about the EFM (Enlightenment File Manager) and have tried to download and compile it many times, but it is not easy at all for the average linux user to do since it's in the early stages. Maybe in the future, you could include a simplyfied guide to compileing and installing the EFM in one of your articles. I would really appricate it a lot. Thanks.

—Graham Greenfield , coderson@inwave.com

---

*The OS*es* of the People*

Pablo Beana wrote ravingly of your “Musings on Linux Profiteering” in the April “Letters”, but he seems to have missed your main point. It's not about money, it's about the fact that “exlusion is one of the main things the free software movement fights against”. Yet here he is, putting down the BSDs because they have a different license, which he either hasn't read or doesn't understand. Unlike Jason Kroll, he forgets that there was free software (including the original BSD, minus a few bits of old AT&T code) long before Linus began coding his first program. Everyone in the Linux community should be glad there are three BSD distributions as well as the hundreds of Linux distro's; there is strength in numbers.

—Ian Darwin, ian@darwinsys.com

Your “Sending Files by Email” article

I must say I was a bit disappointed by the above article in the March 2000 issue of *Linux Journal*.

You mention uuencode as the way to send binary files. Most users these days wouldn't know the uuencode format if they bumped into it on the street—much less when getting an email message comprised of seemingly incomprehensible text.

The standard for sending files by email is of course MIME. You can use metamail for that. This sends any file in MIME format, and every modern e-mail client will understand that.

—Gavrie Philipson, gavrie@netmor.com

The Last Word!

Hi! I usually don't write to the editors of a magazine unless something really gets to me.

The article at the end of each magazine “The Last Word” by Stan Kelly-Bootle is absolutely terrible. I have endured it the last 3 issues when I tried to read it.

Either Mr. Bootle is very intelligent and can say alot of nothing with no practical value or he is just a terrible writer or both.

I don't care about is ego and intelligence (the worlds full of them), But I would like something practicle somewhere. He's wasting your money paying to say nothing!

Thanks for listening!

PS. The picture of him is also terrible!

—Mel Fisher , thecongp@yahoo.com

Your recent article in Linux Journal

Hello Lawernce & *LJ*,

I just wanted to drop you a note and let you know that I caught a few errors in the “Setting up a Linux Gateway” in April 2000 edition of *LJ*.

1) the IP Masq HOWTO hasn't been a Mini-HOWTO for quite some time now.

2) I've found that you don't have to run “modprobe” upon each boot. Then again, I manually load all modules manually to keep system resources to a minimum.

3) The section where you specifically allow/deny internal clients to be MASQed has the wrong netmask. With the shown /24 mask, all clients in the 192.168.0.x network will be able to access the external network. To fix this, it should be changed to a hostmask of “255.255.255.255”. Personally, I like the newer notation of /32.

4) When setting the default route on a client with named machines, you are trusting that DNS is properly configured w/o demonstrating how to set it up. I've seen many times where DNS won't work the first time, etc. I recommend the use of IP addresses when setting the DGW for both reliability and security but this is just my recommendation.

Thats about it. Thanks for helping spread the word of IP Masq to the Linux community!

—David A. Ranch, dranch@trinnet.net

Linux in general

As a real Linux newbie, coming from the Windows world, I am absolutely astounded at the community that has sprung up around Linux and the help that can be recieved just by asking. This does not happen much , if at all, in the Windows world.

As I am just learning, I find your magazine(and another, which I will not mention as I prefer *LJ* quite a bit) extremely helpful in understanding how things work in Linux. As we know Linux is not as easy to understand, being a Unix type OS, it is extremely nice and helpful to find a source where information can be recieved in a relatively understandabe format.

Here's a thought:it might be nice to have a regular(meaning every month) column for newbies and beginners to Linux as it can be very daunting to learn. I put Linux on my desktop mostly because I became sick of Windows crashing every week.

So again ,thanks for the great magazine and keep up the good work. It really does help.

P.S-Tux likes your magazine too!

—Jon A, jonbeth@hotmail.com

Linux Journal Interactive

Thanks for providing this excellent resource.

I was having withdrawal symptoms because my March edition did not arrive. A bit harder to read on the train, but at least I haven't missed out on an entire issue.

—Alan Graham, alang@bigpond.net.au

Re: the Internet in the April 2000 issue

After reading the “impressive” statistics on the Internet's growth in Peter J. Salus' article, and then reading Marjorie Richardson as she “talk[ed] about world domination” in another, I felt compelled to illustrate to your writers some basic truths about the world off-line that they seem to have missed in their zeal.

First of all, over 50% of the world live on less that $2 USD/day, 70% live on less than $10 USD/day and nearly 85% live on less than $20 USD/day. Even given the absolutely cheapest method of getting online (386 with Linux, 1 4.4 modem collected from a dumpster, free ISP),the Internet just isn't a reality for most of the world's citizens. Even if they could afford it, day-to-day survival would push a luxury like the Internet pretty far down the priority list behind the necessities. Add in all the other barriers to access (language, lack of physical access to hardware and infrastructure, culture, anti-US sentiment, anti-technology sentiment, illiteracy, etc.) and

the prospective market for the Internet shrinks even further. Top this off with the knowledge that the rate of impoverishment is increasing rather than decreasing and that the gap between the poor and the rich is widening every day, and you begin to see my point.

Set in the context of the above statistics, it is easy to see how the same data lauded as proof of the Internet's popularity and growth can be re-interpreted in a more realistic light. Instead of looking at the estimated number of current users (260 million) as a huge number, we see it as only 4% of the world's population - the most affluent portion, to boot. The connection of Third World countries starts to seem like something to be derided rather than celebrated, knowing that a wealthy few in those countries will be able to shop at Amazon.com while the vast bulk of the population fight for clean water and a reliable supply of food. Similarly, we see the drop in the Internet's growth rate from 2.2 to 1.5 as the inevitable slow down as market saturation nears rather than the drop as the excited first rush of new users ends.

Perhaps your writers are merely expressing the national tendency to look at the world from a United States-centered perspective (For example, “Americans” live in both North and South America, not just the former). Perhaps they are so thoroughly besotted with technology that they cannot see life proceeding in any other way, despite examples to the contrary appearing nearly everywhere one looks. I personally believe the greatest likelihood is that *Linux Journal* has a vested interest in promoting excitement about the Internet as a means of ensuring its survival. “Without the Internet, Linux would not exist.”, Ms. Richardson states - and neither would her job.

For statistics on world poverty see: http://www.igc.apc.org/globalpolicy/socecon/inequal/gates99.htm

—Matthew Yeo, matthewyeo@netscape.net

Linux business story

My experience is that having Linux solve problems that require stability is a very good start. I had trouble with my CD-recorder - broke 50% of the disks under Win98 - and when one day we had a P200 / 64 MB machine that was too lowly for workstation use, I asked permission to establish a small 100 MBit network and have the machine serve CD-recording. That went without a flaw, and CD recording hardly ever fails.

Then one day we need to take screenshots of some tricky DOS program that escape any real-life solution (HyperSnap, PCXDump, ...). My boss asks if I know a solution, and I suggest VMWare under Linux. Runs fine, we even get to take screenshots of the Win98 install program (not an easy task otherwise).

Later still, we have another P166, only 24 MB ram, to spare. Boss suggests we create a new Linux server for file and printer sharing, and is

willing to put another 32 MB ram into it. We install RedHat 6.1 without X-Windows, and the thing runs so well we never get around to adding the extra ram. That ol' box has yet to have downtime.

We proved that we don't need NT internally, and we're better friends with the Mac folks than we've been for quite a while :)

—Henrik Clausen, HenrikCla@bp.bonnier.dk

AOL Hell May Be Here To Stay

I reference to Mr. Doc Searls' article entitled “Now What: Are We Going t o Let AOL Turn the Net into TV 2.0...” in the April 2000 issue I believe that h e is a little off the mark. Regardless of who designed the Interent it is unfortunate that the powers that be will make sure that they will be able to invade our precious screen space in whatever the future brings. Further more, even though there are those of us out there that find TV ads and other fo rms of advertisement to be intrusive much of the populous doesn't seem to min d their existence. I personally have observed people wanting to watch the ads.

Go to change the channel and they screem bloody murder. Just like the “precious” ad space during the Superbowl half-time. People can't wait to see what the media moguls come up with next to make them want to buy their products.

I would have to say that this is split between to kinds of people. Those who are annoyed by the advertising and those who aren't and honestly want to see the ads. Of course there are the gray areas in between and those who cou ldn't care less either way. So it seems that the people that want to see the a ds are the mindless majority.

Until there is a large enough organized movement against the media moguls they will continue to intrude into your daily life whether you like it or not.

Until that time there is possibly nothing the average citizen or tech sav y individual can do about the problem.

—Jason Bean , beannp@usa.net

---

forum

There was a letter in the April forum that a writer wrote concerning “free” Linux and the superiority of Microsoft OS that I would like to respond to. The first distribution I used was Slackware and I downloaded it disk by disk off the Internet (not all, but what I wanted). I am not a brain surgeon so it was “incredible ” that I found it. Also, I support Windows NT4.0 at work

everyday(I know, what a job). It is clearly not a superior product. Anyone who mentions “highly stable” and NT in the same breath is deluded. Lots

of people use MS because they are locked in (Office97 in one respect). BSD's are not uncommon for this “clearly superior product”. I use Linux at home and I think that is is a better OS.

—randy , rdaniel@surfsouth.com

Thanks for the C-64 Article

Thanks for the C-64 article you did for LinuxJournal April2000 edition.

I still own my beloved C-64 and 1541 5/14 disk drive with a load of games! I bought it back in 1982 and it still runs! I migrated over to the Amiga2000 in 1991 and I still run it and access the web with it.

Although I have been delving into the world of Linux since I can't wait for those corporate bafoons to make up thier minds on when the next Amiga will be out.

I like Linux a lot, it brings me back both the C-64 and the Amiga combined! I love the new discovery of making things work and the inovation of the Linux Kernal and the host of all those free and not so free programs. What joy! But since i work in an environment with WinNT and Oracle and Novell, there are no people to converse with about Linux :(We do have the AtaInta Linux group, but they meet once a month :(. I need to go to school.

Sincerely,

—Israel Cortes , israel@mail.epiclearning.com

Complaint about 3840l1

Listing 3840l1 in the JPython article in the Python supplement looks fine in the file, but the listing in the magazine is misleading. The function 'push' should be defined as:

```
    def push(event): # Callback for regular keys
       display.replaceSelection(event.actionCommand)
```

However, in the magazine it shows up as:

```
    def push(event): # Callback for regular keys

    display.replaceSelection(event.actionCommand)
```

This is illegal Python, because a def line must always be followed by an indented line (unless the entire function is on one line). The reader, trying to figure out what was meant, cannot tell whether the second line is supposed to be the function body, whether the function body is missing (replaced by the blank line in between), or whether the function body encompasses several or all of the following lines.

Please keep this in mind for future listings. Because Python doesn't have braces, we need the indentation to tell where a function ends.

—Mike Orr , info@linuxjournal.com

A letter to LJ

Linux—Way to Go—no, sorry: Linux a Long Way to Go!

I was amused to read how easy it is to send files by e-mail in Unix (*LJ* March 2000, page 120) hust by using command line utilities like sh, tar and uudencode and uudecode. Compare this with the burden Windows users have bear when they select Attach file from their e-mail program's menu.

I am not yet a Linux user and now it seems, it will be some time until I will be. Years ago I tried Yggdrasil's Plung -and-Play Linux, and it did not quite plug and play—instead, it destroyed my hard disk after many a tries in installing it. Now, I decided to try Turbo-Linux evaluation version included with *LJ* (the FREE part printed BIG in the cover, and the evaluation, as usual, in small print...). It turned out just as bad a product. No instructions included, I managed to get the installer going. After many error messages telling aboud accessing past a device's limits, it did not allow me not to configure networking—with no way out but ctrl-alt-del. Perhaps all coild have gone ok if the installer would have told me how to make a special hardware diskette from the image files on the CD. I went to Turbo-Linux's Web site to get some instructions, but when the only documentation turned out to be a two Meg PDF file, I decided to wait some more years until trying Linux ago. I have a hunch, that Turbo-Linux may turn many more people against Linux than towards it... Until I noticed this issue of *LJ* (I have bought many issues before) in a bookstore, I was thinking of buying Corel Linux. but now I think I'll pass Linux altogether.

Regards,

—Matti Vuori , mvuori@koti.tpo.fi

> *I was only showing one way to do it using UNIX commands. Most Linux mail programs, such as mutt, also give you a one button to click on to attach a file. You may do these things in many different ways. I was only pointing one. —Editor*

Comments on "The (not so) Wonderful World of High-Speed Internet Access

In your April 2000 Issue Jason M. Schumaker says regarding ADSL pricing, "Frustration comes from having to pay $60 per month." for a 384K/128K connection. Oh, how that makes my mouth water with envy! In this part of US West-land, the best I can do is an ISDN line (128K if I use both channels) for $100/month. Add $300/month for ISP charges and over

$200/month to get the line to the house and we're talking serious money compared to Jason's paltry $60.

I'd love to pay twice what Jason is paying even for a traditional dial-on-demand ISDN line. I live in northern New Mexico, though, and US West has repeatedly expressed their unwillingness to provide anything approaching modern service here affordably. So much for their ”Life's Better Here“ slogan.

—John McDermott , jjm@jkintl.com

Security problems in PERL/DDB scripts given in Linux Journal March Edition

This morning I was reading your Article in *Linux Journal* (March Edition) about Perl interface to SQL systemes and CGI programming.

Your scripts (i just checked view-ranking.pl, others may be dangereous too) does not verifiy input given by Web user :

```
my $item_id = $query->param("item_id") || 0;

if ($item_id)
{
    # Get (and print) basic information about this item
    $sql  = "SELECT C.category_name, I.item_name, ";
    $sql .= "       I.item_description, AVG(R.rank)";
    $sql .= "FROM RankItems I, RankCategories C, ";
    $sql .= "     Rankings R ";
    $sql .= "WHERE I.category_id = C.category_id ";
    $sql .= "AND   I.item_id = $item_id ";
    $sql .= "AND   I.item_id = R.item_id ";
    $sql .= "GROUP BY I.item_id = R.item_id ";
```

What about if user send something like “12 ; DROP TABLE I” (or other SQL statements) in $item_id ? Depending of the underlying Databse, one can perform completlty arbitrary functions on database and maybe on operating system or at least completly change semantics of SELECT statement.
You should at least test input for meta-chars such as ;,|/\: and escape all input in quotes.

I know these scripts are just examples, but they are read by CGI developpers unfamiliar with good security practices, and can be copied as is in numerous CGI scripts.

I have seen too many enormous security holes during CGIs code reviews made by our company to not mention this. I hope you and editor will include a warning in next issues about potential security issues.

Thanks for reading me.

—Alain Thivillon , Alain.Thivillon@hsc.fr

Hi, Alain. Thanks for writing.

I'm embarrassed to admit it, but I never thought about security issues in the sense that you suggested. I definitely should have, and I will do so in the future.

I'm even more embarrassed that I put an unquoted variable inside of an SQL statement! I thought that I was now using DBI's placeholders consistently in all of my programs—in both my consulting work and in my *Linux Journal* columns—but obviously, some of my old, sloppy programming habits have stuck around.

By using placeholders, which automatically quote variables before submitting the final SQL statement to the database server, it is possible to avoid the sort of security issue that you mention. Given that placeholders can also speed up a query, particularly a repeated query, it's really too bad that I didn't use them more fully here.

Thanks for bringing this to my attention; I'll try to double-check such things when writing future columns.

—Reuven M. Lerner, reuven@lerner.co.il

Microcomputer emulation...

I enjoyed reading your article on game emulation in the April issue of *LJ*. I am one of those who “remember the Spectrums”. I found a great ZX Spectrum emulator that runs on Unices, called XZX. It's homepage is http://www.philosys.de/~kunze/xzx/ If you want to experience with some games, BASIC extenders (the Beta Basic series, etc., pretty much like Simons Basic on C64) and other programs, go to “World of Spectrum”, http://www.void.demon.nl/spectrum.html

Regards,

—Boszormenyi Zoltan , zboszor@externet.hu

---

daemon woman

Marjorie,

Oh please, ever heard of free will?

Lighten up...

Off topic...

The women that spew stuff like you are usually the ones that are not qualified to be a “daemon babe”.

I am glad I let my subscription lapse. The rag (err.. magazine..) gets worse and worse. Now I think I know one reason why.

—“Aaron C. Springer” , a.conrad@ix.netcom.com

> *Ahhh, a person, who doesn't believe women (or men) should be treated as sex objects, can't possibly be able to edit a high-quality technical magazine. I didn't know that. —Editor*

---

a humble request

a picky request, but would it be possible to redesign the spine of *LJ* to put the date at the top, so that when some of us file them in magazine holders, we can see the date and issue right away rather than having to drag out a handful of them first?

thanks.

rday

—rpjday , rpjday@mindspring.com

LJ look and feel

First, let me tell you that the look of *LJ* is vastly improved over years past [has it been years _already_?!] Gone are the bizare sub-heads and some other weird things, but there are still some things that could look better, if you don't mind my [unprofessional] opinion:

1] I am a figure freak, especially graphs and tables: examples from issue #71 are those on pages 54 and 110. Lots of info, but not too pretty and sometimes hard to understand.

2] sub-articles [or whatever you call them] and code listings: I don't mind the pastel colors, but a hairline around the box would be a big improvement. This would also include the 'resources' at the end of articles.

Again, I hope you take these suggestions the right way, and keep up the good work! _Love_ the mag!

—Rafael Block , rafaelb@earthlink.net

Correction in “Artist's Guide to the Desktop, Part 2”

There is a mistake in the “Artist's Guide to the Desktop, Part 2” article in your April 2000 issue of *Linux Journal*.

“The most interesting of the epplets is probably gkrellm, ...” gkrellm is NOT an epplet, but rather an independent application that can be used in any window manager that you may choose.

Otherwise, great work!

—Marius Aamodt Eriksen , marius@linux.com

Subscriber access

As a subscriber to the Economist, I can get all of the current issue in soft copy form. I can't with *LJ*! I guess they aren't as advanced technically as *LJ*.

I subscribed first for Stanley Kelly-Bottle (the main reason I read Unix Review) and for the good Linux stuff.

I know you have a strong geek following, but some of us work from the GUI (I know that's looked down on) and would like to see some of the stuff as entered in things like linuxconf.

Thanks,

—Paul Cubbage , paul@opencountry.net

Feedback

I enjoy your magazine immensely, and look forward to the arrival of each issue. However, I've noticed a disturbing trend in the how-to articles. This trend is to recommend that the reader use linuxconf to configure some option. I'm sure you know that linuxconf is a Red Hat-only tool, and is not available in, say, SuSE or any other distribution.

I would appreciate it if you could make every effort to avoid distribution-specific tools as the answer to configuration options. Being relatively new to Unix, and having a SuSE distribution, I still have no idea what to do to configure Samba daemons to start automatically (April, 2000 issue, page 32) since it simply says use linuxconf.

Thanks for your consideration.

—Tom Wood, tomrwood@email.msn.com

Archive Index Issue Table of Contents

Advanced search

upFRONT

Various

Issue #74, June 2000

The Embedded Linux Consortium and more.

The Birth of the Embedded Linux Consortium: March 1, 2000

Last fall, several companies in the newly emerging embedded Linux market approached me to request that I help create a vendor-neutral Embedded Linux trade association. Initially, I used the resources of my newly established Embedded Linux Portal web site (http://www.LinuxDevices.com/) to propose the establishment of a group called the "Embedded Linux Consortium" (ELC). The idea for the ELC rapidly picked up momentum.

Next, I scheduled an ELC organizational meeting to be held at the Chicago Embedded Systems Conference in March. The purpose would be to create a formation committee to fund the ELC and share responsibility for it until it was incorporated and a proper Board of Directors was elected.

March 1 came, and the ELC organizational meeting drew an overflow crowd—nearly double the expected turnout. We could instantly see that the ELC is destined to be a major force in the embedded market. During the first hour of the meeting, more than $100,000 was pledged to initially fund the organization. Also, Lineo donated the domain name, embedded-linux.org.

A week later, the birth of the ELC was broadcast via a press release that received extensive worldwide coverage (read it at http://www.embedded-linux.org/). The announcement lists the Formation Committee members, summarizes the organization's goals and includes this supporting statement from Linus: "This new Embedded Linux Consortium is an expression of the current explosion of interest in using Linux in thousands of specialized embedded, mobile and appliance applications. The ELC provides a valuable resource in advancing the growing use of Linux in embedded applications, an area where Linux can provide enormous benefit."

Vision and Mission

Although official Vision and Mission statements will be established by the soon-to-be-elected Board of Directors, I indicated my thoughts on the subject in an "ELC Manifesto" posted at LinuxDevices.com and distributed at the ELC organizational meeting:

- Suggested vision: the Embedded Linux Consortium (ELC) will be a nonprofit, vendor-neutral trade association whose goal is the advancement and promotion of Embedded Linux throughout the embedded, appliance and applied computing markets. Members will contribute membership dues and efforts in return for a growing market opportunity for all.
- Suggested mission: to make Linux the number-one operating system of choice for developers designing embedded systems.

In short, by creating and supporting the Embedded Linux Consortium, we will maximize the depth and breadth of penetration of Linux within the enormous —and enormously diverse—embedded market.

Membership Categories

We currently plan to have three membership categories: Corporate Executive Member ($5,000 per year), Corporate Affiliate Member ($1,000 per year) and Individual Member ($150 per year). Recognizing there are many individuals who contribute to the open-source code base which is the basis of Linux itself, I've also proposed that the $150 annual membership fee for Individual Members be waived in the case of individuals who have contributed significantly to the open-source code base.

Activities

The ELC's activities are likely to be along two main threads: technical and promotional. It's likely (although not required) that the Individual Members will be focused mostly on technical activities, whereas promotional and marketing activities will probably be more of a concern of the Corporate Members and, appropriately, funded primarily by them.

On the promotional side, I expect we'll have a marketing task force concerned with evangelizing embedded Linux. Most likely, there will be a web site, PR, trade shows, collateral materials and membership growth.

The technical role of the ELC is less clear than the evangelical role, in light of the enormous success the existing open-source development process has had in bringing Linux (and its related technologies) to where they are today.

Nonetheless, it is certainly possible that technical committees or special interest groups will coalesce around issues of interest to multiple ELC members. Whether these translate into standards activity remains to be seen. In any case, the existing Linux and open-source development process must be supported, not circumvented or undermined.

Status and Further Information

The ELC is now officially incorporated as a non-profit trade association in the state of California, and has the beginnings of a web site in operation. For the latest ELC news and info, or to obtain a membership application, visit http://www.embedded-linux.org/.

—Rick Lehrbaum

Interview with Brian Behlendorf

Brian Behlendorf is one of the chief software architects behind the Apache project and is the instigator of SourceXchange. This project provides a method for programmers to get paid while writing open-source software. As such, we would like to see it succeed. We talked with Brian by e-mail at the end of March to see how things were progressing.

Margie: SourceXchange started as a project of O'Reilly and HP, and then was spun off as Collab.Net. What was the reasoning behind this move?

Brian: I incubated the ideas behind the company while employed by O'Reilly, and worked with people there to develop a business plan, which we then shopped to a couple of investment firms. After selecting Benchmark, we spun the company out into its own entity, and I moved to CTO when we found a CEO.

It was always the plan that I would be working on some ideas at ORA, and then we'd spin them out when there was sufficient interest. There was much more interest much sooner than we thought there would be. SourceXchange is one service of Collab.Net—there will be more.

Margie: Do O'Reilly and HP still participate?

Brian: Yes. Tim O'Reilly is on our board, and we have lots of communication with ORA. HP is still involved as a sponsor in SourceXchange and is also involved in some of our additional products. They are the first customer of our open-source project hosting infrastructure, on which their new web site (http://www.e-speak.net/) and open-source project sit.

Margie: Has the change proved of benefit to the project?

Brian: Becoming our own company has helped us hire the people we needed to, and focus more deeply on this as a business, which we feel it needed to be long-term.

Margie: Tell us exactly how SourceXchange works.

Brian: Well, the http://www.sourcexchange.com/ site goes into quite a bit of detail on this, so I won't repeat what's said there. The short answer is:

> Sponsor submits an RFP (request for program)-->Developers submit proposals/bids on that RFP-->Sponsor selects a proposal.-->Project begins.-->Developers work towards a milestone, then upload their code.-->Sponsors review it and approve or disapprove.-->If they disapprove, developer keeps working until the sponsor is happy, or until a peer reviewer says the milestone is met.-->Repeat until all milestones are approved and project is completed.-->Sponsors, developers and peer reviewers rate each other.
>
> Repeat. =)

Margie: How many programming projects have actually been finished? Which ones?

Brian: Two—the Generic Data Packet Recorder, sponsored by Sparks.com and completed on 2/18/00; and a Test Suite Framework for the Apache Web Server, sponsored by HP and completed on 11/20/99. For details, see www.sourcexchange.com/ProjectBrowse?Button=Search&browse_status=Completed&browse_status=Archived.

Margie: What is the biggest project to be finished at this time?

Brian: Both of them were $5,000 US projects, but there are now some in the $25,000 US range in the system.

Margie: Which project have you personally been the most interested in?

Brian: The test suite for Apache was fairly interesting to me. Right now, the most interesting one is probably the Apache 2.0-related handler (Listener Module) for a protocol called BXXP, sponsored by Invisible Worlds (project #13). I've also got a project of my own I'm pretty happy about, a Java servlet- and WebMacro-based browser for UNIX mbox-format mail archives, sponsored by Collab.netproject (project #11).

Margie: Does SourceXchange take up all your time, or do you still get to do some programming?

Brian: Collab.Net in general, as well as doing all the political and back-end work for Apache, consumes most of my time. I haven't done serious programming in a long time, but still write the odd Perl script each week.

Margie: Obviously, you are still involved with Apache. What's going on there?

Brian: Tons—ApacheCon was just completed and was a resounding success, with over 1000 attendees and some great sessions. Apache 2.0 is in alpha. XML and Jakarta are taking off.

Margie: Do you feel the SourceXchange project is a success? (Are programmers actually getting paid?)

Brian: Yes. There is over $300,000 in the system right now for developers, and it's just the beginning, really.

Margie: Do you see this project as leading the way for how all programming will be done in the future?

Brian: No, but it will be an important component. Sometimes you are willing to trade off quality and correctness for determinism and speed, and thus you need in-house developers. Sometimes the software you're developing needs to be done in-house for proprietary reasons. I'm not one to state that all software will be free in the future, although I think most of it will be.

Margie: What do you see in the future for open-source software and Linux?

Brian: Great things. Further expansion. As software becomes infrastructure, it becomes commoditized, and open-source software is the most stable state of infrastructure because of its ubiquity and low cost to use.

Margie: Anything you'd like to add?

Brian: Not really; I've done an awful lot of proselytizing over the last few years, and now feel a need to turn that energy toward making sXc and other Collab.Net projects a success, so I've been more quiet recently.

Margie: To wrap up, how about some personal-type information?

Brian: I've been married for almost five years, have a house in San Francisco, am really into DJ'ing ambient and dub music, use a Sony Vaio C1-XS as my laptop, drive a Jeep, and enjoy microbrew root beer.

Margie: Thanks very much for taking the time to answer our questions. -- Marjorie Richardson

LJ INDEX—JUNE 2000

1. Percentage of Linux developers who don't care if the tools they use are open source or proprietary: **87**
2. Number of tool categories (out of 11) judged as adequate by 75% or more of developers: **2**
3. Percentage of Linux developers using predominantly command-line tools and utilities: **50**
4. Spending on Linux among the top 100 financial institutions in 1998: **$50 million US**
5. Projected spending on Linux among the top 100 financial institutions in 2003: **$200 million US**
6. Yearly growth rate between the above: **32%**
7. Number of lunch hours it took Dr. Linas Vepstas to install Linux on an IBM 390 mainframe: **1**
8. Number of Linux machines that can be administered by an IBM 390 mainframe running the VM operating system: **50,000**
9. Number of sites surveyed by Netcraft in March 2000: **13,106,190**
10. Percentage of sites running on Apache: **60.05**
11. Total number of sites running on Apache: **7,870,864**
12. Gain by Apache in the most recent month: **1,388,156**
13. Percentage of sites served by Microsoft IIS: **20.9**
14. Loss in Microsoft server share percentage from prior month: **1**
15. Number of press room computers at LinuxWorld/Spring running Windows: **9**
16. Number of press room computers at LinuxWorld/Spring running MacOS: **1**
17. Number of press room computers at LinuxWorld/Spring running Linux: **0**

18. Number of press room computers at LinuxWorld observed with Slashdot on the screen at the same time: **6 or 60%**
19. Number of people killed by sharks each year: **10**
20. Number of sharks killed by people each year: **60,000,000**

Sources

- 1-3: The Linux Show
- 4-6: Evans Marketing Services
- 7-8: IBM, Tower Group research (sourced by IBM)
- 9-14: Netcraft
- 15-18: Doc Searls
- 19-20: David Siegel

THEY SAID IT

“I'll say this about Linux: it's the first time I've seen UNIX on the right platform.”

—Steve Ballmer, Microsoft

“No marketing organization can withstand the effects of a community which generates an ever increasing number of effective advocates.”

—Russ Pavlicek, Compaq

“There's an opportunity for China to play a significant role in the Linux world... That certainly could allow China to take its place on the world stage as a software-producing country.”

—Dan Kusnetzky, International Data Corp.

“I'm glad IP isn't IP.”

—Dan Lynch, in a meeting about the Internet and patents

SAY HOW?

In April, Netcraft reported that over half of the exhibitor companies at the last Linux Expo in London were running Microsoft IIS web servers (which runs on Windows 95, 98 and NT) rather than Apache or some other appropriate server on Linux. Among other interesting discrepancies, Netcraft noted these:

- Compaq.com was running Solaris until enough Compaq people noticed, so now compaq.com runs Tru64 UNIX.

- linux.ora.com runs Solaris.
- Early adopters of Windows 2000 include linuxbeacon.com, linuxanswers.co.uk, slashdot.org.uk and freshmeat.org.

(Note: the last two are parody sites.)

Other items:

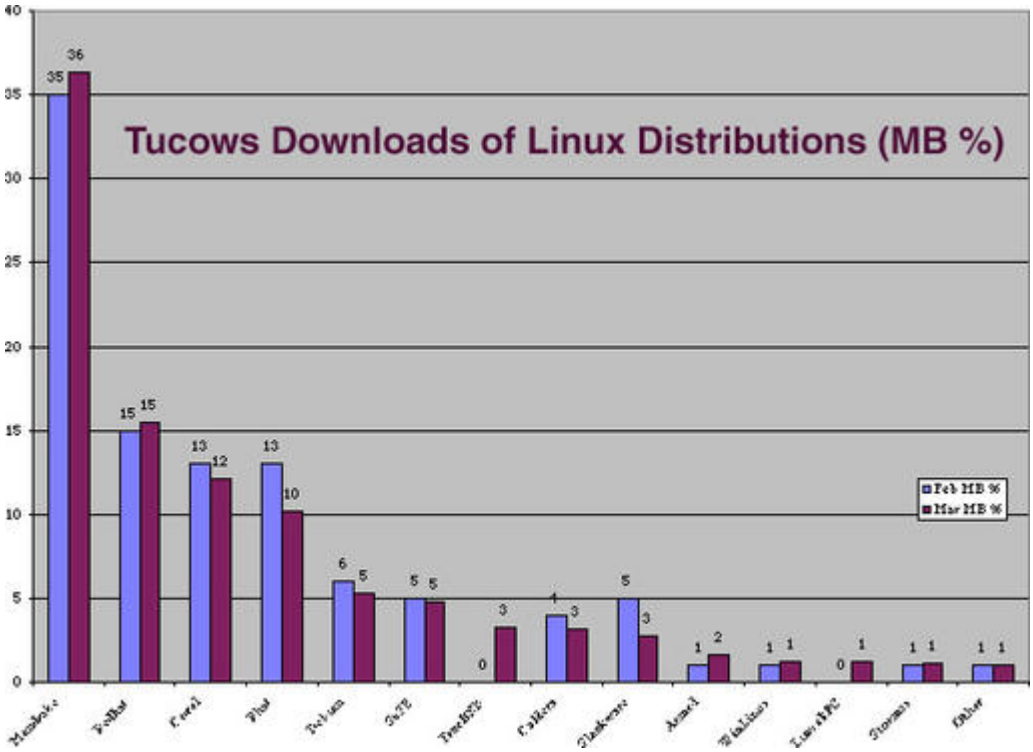
- The microsoft.eu.org home (another parody site, at bero.exit.de/training/mcle.html) runs on Linux.
- linuxsucks.org runs on Linux.
- yahoo.com runs Apache on PalmOS.
- Apple's many servers run on a combination of Solaris, Linux, MacOS, Mac OSX (a BSD variant, still pre-release) and AIX—no Microsoft.
- While dell.com runs Apache on Linux, most of its other sites run IIS/4 on Windows NT4/98.
- HP runs a combination of Linux, Solaris, HP-UX and NT4/98
- IBM is mostly AIX, with some Linux, NT4/98 and OS/2.
- Intel is mostly running IIS/4 on NT4/98 (although the very personal pentium.net runs Apache on Linux).

Check for yourself—the "What's that site running?" page at Netcraft, http://www.netcraft.com/whats/.

—Doc Searls

MANDRAKE MAGIC CONTINUES

Mandrake held its number-one position in total downloads (measured in MB) from Tucows in March. Even though it's down a couple of points from its January high, it still accounts for over a third of the downloads total. Red Hat held even with 15% at #2. Corel slipped a point to 12% in the #3 spot. Phat Linux dropped three points, but still held the #4 position at 10%. Debian and SuSE follow with 5% apiece. The rest are between 3% and 1%.



Tucows Download Chart

THE BUZZ

What were Linux people talking about in March and early April? Below is a sampling of some of the hotter news stories over the past few weeks, as reported in “The Rookery”, *Linux Journal*'s on-line source for news, notes and reports from the field (updated daily and found at our web site http://www.linuxjournal.com/):

- Federal court ruling that Microsoft violated anti-trust law, and the ensuing class-action suits filed against Microsoft. Also, discussed quite elegantly in Bryan Pfaffenberger's article “Guilty! What's Next for Linux?” (http://www.linuxjournal.com/articles/currents/018.html).
- Sixth Circuit Court's finding that source code is a form of expressive, as well as functional, speech.
- IBM's dumping of Red Hat stock. What's that about?
- Caldera's IPO, disappointing or realistic, seems to be a matter of who you talk to.
- W. R. Hambreck & Co. giving a “buy” ranking to VA Linux.
- InterVideo's claim to have the “first legal DVD software solution for the Linux OS.”
- The coming showdown between Linux and Windows in the embedded systems market.

STUPID PROGRAMMING TRICKS-->Raster Bars and Console Graphics

Hello, and welcome yet again to another episode of Stupid Programming Tricks! Last month, which I hope you didn't miss, we got a wee bit complicated, making a sine scroller by plugging sine functions into each other. The process involved lots of variables and functions, and really taxed the processor. This month, it's time to relax a bit from complicated things, but nevertheless we'll do something cool, and it *will* involve sines.

Former demo-sceners may have noticed a trend in this column: that much of what we do is passed down from a certain tradition. Where is the 3-D <\#224> la Quake? Where are the filled vector graphics? Where are the trippy acid effects? Alas, your amiable author missed the PC scene for a short-lived adventure in, well, something else, before discovering Linux, so we still believe a demo should have a scrolltext and that 2-D is *okay*. Now, we've done more with scrolltexts than most people can stand; however, there is a tradition just as ubiquitous as the scrolltext, and probably even easier: the raster bar.

A raster graphic is basically a bitmapped graphic (as opposed to a vector graphic), thus a raster bar is a bitmapped graphic of a bar. Usually we draw these bars with cool, shaded colors, so that they look like horizontal pipes or laser beams, and we make them bounce up and down on the screen, which looks rather cool. What is the point of such bars? There is none! Well, actually, sometimes it's to show off the computer's colors; sometimes to delineate different areas of the screen; sometimes it's just to add something interesting and technological to look at. When the palette is properly set up, the bars can look like shiny metallic pipes or perhaps glorious rainbows. You can even scroll the colors inside of raster bars for all kinds of funky effects. You could sync the color scrolling to the vertical position of the bar, as though the passing raster bar uncovered hidden colors in the screen. You could cycle the colors in numerous ways inside the bar (turning blue pipes to purple, red, orange, yellow, green and back to blue), or you could pass colors between several raster bars on the screen.

Another clever trick is to include a scrolltext *inside* of a raster bar. This has numerous advantages; for one thing, you don't have to worry about making sure the background shows up properly in the black space underneath the letters, since you're filling the whole thing up with raster color. You don't need to clean up areas before writing with raster bars, because the bars overwrite whatever's there initially. And, the colors of the raster bar can highlight the text like a high-energy laser beam.

One influential demo for me on the Amiga was Hot Copper by Acro of Rigor Mortis. It used several large raster bars of different colors scrolling up the

screen and then falling down again in order to display several different scrolltexts of varying speeds with different text. In the days before the Net as we know it, words were rare and meant much more; scrolltexts were like messages in bottles found at the beach, greetings from faraway shores, by people you actually knew! How did the text travel so far? Who passed what disk to whom at what party, how many boards did it go through, how long until it got to you? If only Amiga emulation weren't so harassed by copyright zealots, we'd have a more open time enjoying these (the Amiga ROMs are still copyrighted, but at least the demos aren't). Alas, I degenerate.

We're going to make six bars, each 320 pixels wide by 17 pixels high. Each bar will get to use nine colors, running from darkest at the top to lightest in the middle and dark again at the bottom (that's eight colors which get used twice in each bar, with one bright color right in the middle). This makes the shiny, pipe-looking effect for our bars. In honor of Roy G. Biv, we'll have red, orange, yellow, green, blue and purple bars, even though logically we should use red, yellow, green, cyan, blue and purple. Of course, cool Commodore coders made rainbow bars, but they're too cool for us. Still, we *are* fairly daring in our own right; after all, we're using “runs-as-root-can-really-destroy-your-system” SVGAlib. Let's look at our algorithm.

Step one is to draw the bars. We'll make six arrays to store our six different bars: redbar, orangebar, yellowbar, greenbar, bluebar and purplebar. We could dynamically allocate the memory, but why bother, since we already know the size of our graphics? Well, it's in case we change graphic size, but let's be lazy this time around. First, we have to set the palette properly, so we'll do nine shades of red, nine shades of orange, nine shades of green, nine shades of blue and nine shades of purple. A simple loop fixes this, rendering 54 colors total, starting at palette color #192, because it's nice to save the earlier palette colors in the event we add something else. Once we've set the palette, we draw the red bar in red color, store it in the redbar array, then loop through and do the same for the other colors. When we're finished, we'll have six bars in memory, ready to be blitted wherever we like!

Step two is to draw the bars on the screen in our never-ending loop of joy. We'll set a loop that increments a variable called X, needed because the vertical positions of our bars will be based on a sine function of X, shifting the phase a bit for each bar so that we get a wave-like effect (instead of having every single bar on top of the other). We'll draw the bars, starting with purple and working toward red, hold the screen still for a vertical refresh (usually 1/60th of a second), then clear the screen and repeat the loop until someone exits.

Raster bars, alas, have little practical value; in fact, they're an epitomic cliché, but I sincerely hope we can continue this cliché into the 21st century and

beyond. Perhaps we should put raster bars in rocket ships and send them out into the galaxy as evidence of our civilization...

—Jason Kroll

Listing. Making Raster Bars, raster.c

```
// gcc -Wall -O2 raster.c -lvgagl -lvga -lm
#include <vga.h>
#include <vgagl.h>
#include <math.h> /* sines! */
#include <stdlib.h> /* malloc */
#define GMODE G320x200x256
GraphicsContext *virtualscreen;
GraphicsContext *physicalscreen;
{
  char *redbox, *orangebox, *yellowbox;
  char *greenbox, *bluebox, *purplebox;
  char c, d; /* to keep track of color */
  short int x; /* a counter for our sine */
  vga_init(); /* start svgalib */
  gl_enableclipping(); /* just in case */
  vga_setmode(GMODE); /* set mode */
  gl_setcontextvga(GMODE);
  physicalscreen = gl_allocatecontext();
  gl_getcontext(physicalscreen);
  gl_setcontextvgavirtual(GMODE);
  virtualscreen = gl_allocatecontext();
  gl_getcontext(virtualscreen);
  /* Allocate the memory for our bars! */
  redbox = malloc(WIDTH*17*BYTESPERPIXEL);
  orangebox = malloc(WIDTH*17*BYTESPERPIXEL);
  yellowbox = malloc(WIDTH*17*BYTESPERPIXEL);
  greenbox = malloc(WIDTH*17*BYTESPERPIXEL);
  bluebox = malloc(WIDTH*17*BYTESPERPIXEL);
  purplebox = malloc(WIDTH*17*BYTESPERPIXEL);
  /* set up the palette! */
  c=0; /* red */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       7*d, 0, 0);
  c=1; /* orange */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       7*d, 3.5*d, 0);
  c=2; /* yellow */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       7*d, 7*d, 0);
  c=3; /* green */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       0, 7*d, 0);
  c=4; /* blue */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       0, 0, 7*d);
  c=5; /* purple */
  for (d=1; d<10; d++)
    gl_setpalettecolor(191+c*9+d,
                       7*d, 0, 7*d);
  /* Now is when we draw the bars! */
  for (d=0; d<9; d++) {
    gl_hline(0,d,WIDTH-1,192+d);
    gl_hline(0,16-d,WIDTH-1,192+d);
    gl_hline(0,17+d,WIDTH-1,201+d);
    gl_hline(0,33-d,WIDTH-1,201+d);
    gl_hline(0,34+d,WIDTH-1,210+d);
    gl_hline(0,50-d,WIDTH-1,210+d);
    gl_hline(0,51+d,WIDTH-1,219+d);
    gl_hline(0,67-d,WIDTH-1,219+d);
    gl_hline(0,68+d,WIDTH-1,228+d);
    gl_hline(0,84-d,WIDTH-1,228+d);
```

```
      gl_hline(0,85+d,WIDTH-1,237+d);
      gl_hline(0,101-d,WIDTH-1,237+d);
    }
    /* copy the bars to our arrays */
    gl_getbox(0,0,WIDTH,17,redbox);
    gl_getbox(0,16,WIDTH,17,orangebox);
    gl_getbox(0,34,WIDTH,17,yellowbox);
    gl_getbox(0,51,WIDTH,17,greenbox);
    gl_getbox(0,68,WIDTH,17,bluebox);
    gl_getbox(0,85,WIDTH,17,purplebox);
    /* the main loop. This severely and
     * inefficiently burns CPU but we're
     * feeling sadistic on the chip today.
     * 12 raster bars is a lot, so comment
     * out a few if your computer grinds
     * to disastrous effect.
     */
    for (x=d=0; d==0; d=vga_getkey()) {
      x++;
      gl_putbox(0,40*sin((x+0)/24.0)+40,
                WIDTH, 17, purplebox);
      gl_putbox(0,40*sin((x+10)/24.0)+40,
                WIDTH, 17, bluebox);
      gl_putbox(0,40*sin((x+20)/24.0)+40,
                WIDTH, 17, greenbox);
      gl_putbox(0,40*sin((x+30)/24.0)+40,
                WIDTH, 17, yellowbox);
      gl_putbox(0,40*sin((x+40)/24.0)+40,
                WIDTH, 17, orangebox);
      gl_putbox(0,40*sin((x+50)/24.0)+40,
                WIDTH, 17, redbox);
      gl_putbox(0,-40*sin((x+0)/24.0)+144,
                WIDTH, 17, purplebox);
      gl_putbox(0,-40*sin((x+10)/24.0)+144,
                WIDTH, 17, bluebox);
      gl_putbox(0,-40*sin((x+20)/24.0)+144,
                WIDTH, 17, greenbox);
      gl_putbox(0,-40*sin((x+30)/24.0)+144,
                WIDTH, 17, yellowbox);
      gl_putbox(0,-40*sin((x+40)/24.0)+144,
                WIDTH, 17, orangebox);
      gl_putbox(0,-40*sin((x+50)/24.0)+144,
                WIDTH, 17, redbox);
      gl_copyscreen(physicalscreen);
      gl_clearscreen(0);
      vga_waitretrace();
    }
    return 0; /* it's almost a tradition! */
  }
```


Installing Window Maker by Michael Hammel is a tutorial on the basics of getting started with the Window Maker window manager. It is an excellent companion to the third part of his “Artists' Guide to the Desktop, Part 3” in this issue.

Data and Telecommunications: Systems and Applications is a book review by Derek Vadala.

Archive Index Issue Table of Contents

Advanced search

Just Folks

Peter H. Salus

Issue #74, June 2000

Who remembers the “gang of five” who started USENIX?

Who did what concerning Linux 1.0, and who's who in open systems, are intriguing queries. Not merely because I think “open” goes back 45 years, but also because I'm not at all certain that any of today's contemporaries (other than Linus, Guido, Larry and Eric) will be considered important in a decade. While Dennis Ritchie and Ken Thompson are remembered, other influential people have faded into the mists of time.

This isn't meant to dismiss the activities of, say, Jon “maddog” Hall. But who remembers the “gang of five” who started USENIX? Who recalls that Lou Katz, then at Columbia University, was the first President? Or that Katz and Reidar Bornholdt organized the very first UNIX Users Group meeting (May 15, 1974, in the Merritt Conference Room at Columbia's College of Physicians and Surgeons)?

Nor do I want to devalue Tim Berners-Lee's contribution. But Peter Deutsch's archie has vanished, as have gopher, veronica, jughead and Mosaic.

We talk about “Internet time”, but our memory of individuals is even worse than our memory of things. So I thought I'd put together a different sort of merit-based who's who—or perhaps a hall of fame for people whose work has had a truly lasting impact. (Note, I intentionally omitted the obvious: Torvalds, Ritchie, Stallman, etc.)

- Eric Allman for Sendmail (1978)
- John Backus for creating FORTRAN (1957)
- Gordon Bell for the PDP-4 through PDP-8 and the VAX
- Fernando Corbato for writing CTSS, the Compatible Time-Sharing System (1963)

- Edson de Castro for collaborating with Bell
- Steve Crocker for inventing the form and writing RFC 1 (1969)
- Ralph Griswold for SNOBOL4 (1971) and ICON (1983)
- Brian Kernighan for being the “K” in K&R and in AWK
- Don Knuth for TeX and *The Art of Computer Programming*
- Mike Lesk for uucp, grep, lex, tbl and refer
- J.C.R. Licklider for making the ARPANET possible
- John McCarthy for LISP 1.5 (1962)
- Doug McIlroy for the idea of pipes
- Bob Metcalfe for Ethernet
- John Ousterhout for Tcl/Tk
- Jon Postel for running IANA
- Bjarne Stroustrup for C++
- Andy Tanenbaum for MINIX
- Larry Wall for rn, patch and Perl

I guess I could easily make this list twice as long. But with very few exceptions, the names would be even less familiar than the ones above.

However, if Backus and his team hadn't developed FORTRAN, we wouldn't have COBOL or Algol or most other languages we use (such as C and PASCAL, both 1971). Without MINIX, we'd have no Linux; without the ARPANET, no Internet.

My real point isn't that everyone ought to memorize who did what to whom. I'm not sure if it matters at all. Alexander Graham Bell, Rudolf Diesel, Tesla and Marconi have their places in technological history, but hardly anyone except historians of science remembers Wankel or even Stephenson. Thus, who's-who volumes are highly ephemeral: some of those once thought important will fade away in a few years. My guess is that we are like those “trunkless legs of stone” in “Ozymandias”.

Peter H. Salus , the author of *A Quarter Century of UNIX* and *Casting the Net*, is an *LJ* contributing editor. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

The New Beginning

Doc Searls

Issue #74, June 2000

“The word, in the end, is the only system of encoding thoughts—the only medium—that is not fungible, that refuses to dissolve in the devouring torrent of electronic media.” --Neal Stephenson, In the Beginning Was the Command Line

The word, in the end, is the only system of encoding thoughts—the only medium—that is not fungible, that refuses to dissolve in the devouring torrent of electronic media. --Neal Stephenson, In the Beginning Was the Command Line

Jokes beginning “If operating systems ran airlines” first appeared on the Web around 1995. Linux wasn't mentioned back then, but it is now, like in this set published in the September 1999 *Linux Gazette* and condensed here:

- UNIX Airways: everyone brings one piece of the plane along when they come to the airport. They all go out on the runway and put the plane together piece by piece, arguing non-stop about what kind of plane they are supposed to be building.
- Mac Airlines: all the airline personnel look and act exactly the same. Every time you ask questions about details, you are gently but firmly told that you don't need to know, don't want to know, and everything will be done for you without your ever having to know, so just shut up.
- Windows Air: the terminal is pretty and colorful, with friendly stewards, easy baggage check and boarding and a smooth take-off. After about 10 minutes in the air, the plane explodes with no warning whatsoever.
- Windows NT Air: just like Windows Air, but costs more, uses much bigger planes, and takes out all other aircraft within a 40-mile radius when it explodes. (The 1995 version—everyone marches out on the runway, says the password in unison, and forms the outline of an airplane. Then they all sit down and make a whooshing sound like they're flying.)

- Linux Air: disgruntled employees of all the other OS airlines decide to start their own airline. They build the planes, ticket counters and pave the runways themselves. They charge a small fee to cover the cost of printing the ticket, but you can also download and print the ticket yourself. When you board the plane, you are given a seat, four bolts, a wrench and a copy of the Seat-HOWTO.html. Once settled, the fully adjustable seat is very comfortable, the plane leaves and arrives on time without a problem and the in-flight meal is wonderful. You try to tell customers of the other airlines about the great trip, but all they can say is, “You had to do *what* with the seat?”

I'd change that last one to open with “UNIX geeks who finally figured out what kind of plane they were supposed to be building ...” And I'd end it with, “After a while, nearly everyone flew Linux Air, whether they knew it or not.”

To the extent that flying with computers happens mostly over the Net, the Linux takeover is already underway. The recent surge in Apache's dominance owes to the growing adoption not just of Linux, but of the BSDs and other UNIX breeds that share Linux' virtues.

But there's still that problem with the seats. I witnessed it at PC Forum in March, where attendees were given Net access (and everything else they could figure out) through dozens of VA Linux systems running GNOME. This was candy for the technical types, but it was like chewing on rubber bands for everybody else. Under normal conditions, I'm about the last guy you'd want to call for tech support, but I found myself walking around explaining things to venture capitalists, dot-com zillionaires and other smart guys who looked as lost as moths in lampshades, flummoxed by a user interface that produced cryptic pop-out menus and windows that iconized to locations other than the bottom of the screen.

“How can I open a Word file?” one guy asked. Another looked for “Excel or something like it”. We found ways to help some of these folks, but the lack of obvious “office” applications did not sell Linux as a desktop OS. I asked more than a dozen of these guys if they were tempted in any way by their experience with Linux at the show. Without exception, the answer was no (but a few open-minded tech types had kind words for GNOME).

The break-out session didn't help, either. Despite valiant efforts by the VA guys, it was too easy for the majority to dismiss the evidence: some Mozilla innovations (still a no-show), a nice little open-source presentation program (with jaggy screen fonts) and new banner-blocking software (old hat on Windows). Afterwards, a veteran *PC Magazine* editor told me they really like Linux and want to give it positive coverage, but in the absence of productivity

applications, there's almost nothing other than the usual sports coverage of server-vs.-server games.

Nobody is more aware of the situation than Larry Augustin, the founder, President and CEO of VA Linux. In February, Larry and I were both on an open-source panel put on by the New York New Media Association. There, he made the surprising point that so far, open-source developers haven't understood the significance of Microsoft Office, which—far more than Windows—is at the core of Microsoft's appeal. After we got back, I asked him to run that one by me again to make sure I got it right. He replied:

> Open-source developers understand UNIX. This is part of what made it possible to create a better UNIX—Linux. In order to create a better MS Office, open-source developers need to understand MS Office in as much detail as they understood UNIX. My fear is that the open-source developer community doesn't understand Office. It can't create what it doesn't understand. What we need are more developers using Windows and Office.

Tall order. Most open-source developers I know would rather not stain their retinas with light from Microsoft pixels. So we live with office suites (including fractions of suites) from Sun (StarOffice), Applix and Corel. All three are closed source. In the absence of source code, there's little for developers to work with, so the problem persists.

Maybe we should come at it from a different angle. Airlines are kind of a server-like metaphor. Let's try a metaphor that works better for clients: cars. That's what Neal Stephenson does in his outstanding new book, *In the Beginning was the Command Line*. Best known for his bestsellers *Snow Crash* and *Cryptonomicon*, Stephenson is (like me) a convert to Linux from the MacOS. To frame his context, he offers an automobile metaphor:

> Microsoft—Started out selling three-speed bikes (MS-DOS). These were not perfect, but they worked, and when they broke, you could easily fix them. Eventually, they came out with a colossal station wagon (Windows 95). It had all the aesthetic appeal of a Soviet worker housing block, it leaked oil and blew gaskets, and it was an enormous success. NT was an off-road version: no more beautiful than the station wagon, and only a little more reliable. Apple—another bike dealership that one day began selling motorized vehicles, expensive but attractively styled cars with their innards hermetically sealed, so that how they worked was something of a mystery.

> Be—a dealership selling fully operational “batmobiles”. These were more beautiful and stylish than the Euro-

> sedans, better designed, more technically advanced, and at least as reliable as anything else on the market —and yet cheaper than the others.

> Linux—not a business at all. It's a bunch of RVs, yurts, tepees and geodesic domes set up in a field and organized by consensus. The people who live there are making tanks. No ordinary tanks, these. They've been modified in such a way that they never, ever break down, are light and maneuverable enough to use on ordinary streets... Best of all, these tanks are being cranked out, on the spot, at a terrific pace, and a vast number of them are lined up along the edge of the road with the keys in the ignition. Anyone who wants to can simply climb into one and drive it away for free. These are sold by volunteer hackers with bullhorns: “Save your money! Accept one of our free tanks!... We will send volunteers to your house to fix it for free while you sleep!” The buyers reply, “Stay away from my house, you freak!” and “Can't you see that everyone is buying station wagons?”

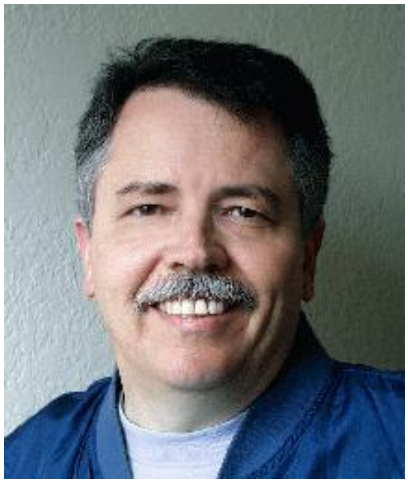
The question is, for how long? I see an answer in the original Volkswagen Beetle. Ugly, uncomfortable, noisy and lacking conveniences like AC and automatic anything, VW bugs became a sensation in the early sixties for three simple reasons: they were cheap, reliable and easy to fix. In fact, they were so easy to fix that I remember a day when a buddy and I took out an engine and put it back three times. I'm not even sure we were sober.

On the server side, we have something of a VW Bug in the Cobalt Qube, a cute little appliance that's handy for the SOHO (home office) market. On the client side, we'll soon see appliances from Intel and others. But there's a big problem with the appliance concept: they're closed as bricks, almost by definition. That disqualifies them as VW Bugs.

The real equivalent of a VW Bug is a cheap and charmingly ugly client box that can run a stripped-down open-source office suite and bring up a bash shell. While that may sound scary to GUI-<\#252>ber-alles bigots (like Neal Stephenson and I used to be), consider this item: even though Apple will still have heaps of closed stuff running on top of their OS-X kernel (Mach inside a custom version of BSD—all open source), the OS will be accessible where it counts, through shells and a command-line interface. This is a far more interesting and useful innovation than the pixel job Apple is giving its old interface. Why? Because its market includes mechanics as well as drivers. No mechanics—no market.

Today, Microsoft is to operating systems what Detroit was to automobiles in 1963. They're way too comfortable making unreliable chrome-encrusted land yachts that people buy out of habit and fear. All it's going to take to break those

habits is a box that both drivers and mechanics would love. Remember, never underestimate the fear-reducing powers of a first-rate mechanic—especially when it's yourself.



Doc Searls (info@linuxjournal.com) is Senior Editor of *Linux Journal* and co-author of The Cluetrain Manifesto.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #74, June 2000

Our experts answer your technical questions.

Route 675

I am setting up a DSL and my topology is simple: Linux box connects to hub, which connects to 675, which connects to the DSL carrier. I put a Win95 machine in parallel (it also connects to hub) and it connects fine to the DSL's DNS and the Linux machine. The Linux machine sees the Win95 machine and also the 675 router's IP. The Linux machine does not see beyond the router (the Cisco 675) to the DNS, nor can the Web connect to it, though it should be able to using static IPs. I think this is an issue with this router and Linux; in the documentation there is a note: “must have termcap database installed on Linux or Cisco 675 may not work correctly.” I do have termcap installed (I did a full SuSE install). Do I have to “run” termcap somehow, as an executable program, even though it is “installed”? (I tried to **rpm** the termcap.rpm and it said “already installed”.) Or is termcap already running automatically?

Cisco was kind enough to tell me they have a contract with USWest which precludes them from customer support for the 675. The quote in full from the Cisco manual: “Computers running Linux without the term/termcap database installed will have trouble connecting to Cisco equipment. The message BAD ADDRESS is sometimes displayed as an error message. The user can install the term/termcap database from his Linux install disks/CD.”--Scott Cameron, lscott11@uswest.net

It sounds as if your network setup on the Linux machine is missing a default router (or gateway) in the configuration. If you type **route -n**, you will see a list of the routing table that the kernel maintains. The default route is the one the kernel will choose for network packets in the event no other paths are possible to the destination. In addition to having a default route, you will also need to have DNS set up properly or you won't be able to resolve domain names into IP

addresses, which is how network communication happens. Thus, if you can ping the IP address of your DSL router and no further, then it is likely you are missing a default route. If you have a default route defined (it should be the 675 router) and you are unable to ping a site such as www.linux.org, then DNS is probably not configured properly. About the termcap statement: termcap is a library that provides a database of the terminal capabilities of various terminal emulations. It is not required to use the 675 for networking purposes. However, if you need to access the embedded operating system on the router, you can telnet to it (or use a direct serial connection—I'm guessing because I have never seen a 675) and access its built-in features. This is where the termcap would be required, as it will use some type of terminal emulation for which your terminal will need to know the mappings for what the character sequences do. So, it is necessary only if you intend to log in to the router and change the settings. —Andy Bradford, andyb@calderasystems.com

Li Lo, Sweet Linux

I am new to Linux, just three issues into my first *LJ* subscription and have purchased several informative books. I've installed Linux on my C drive along with Windows 98. My Linux version is Red Hat 6.0. It loaded LILO to provide direction to Linux and W98. Now I wish to install DOS. One of your recent issues provided information on how LILO is to be modified, but not on how to access LILO or where it is located. Frankly, I am confused. On one recent attempt to load Linux, which I aborted, I discovered that LILO was not erased, even when I completely reformatted the Linux partition. So, how do I find the LILO program to change it so DOS can be included? —B.E. (Gene) Johnson, gene-bj075@msn.com

LILO is configured through the /etc/lilo.conf file. Actually, if you install Linux on a PC that already has Windows 98, you should first have enough space on your disk(s) to hold Linux. You need to defragment the Windows disk, then you may have to repartition it. There is a utility called FIPS on your Linux CD which allows repartitioning a disk without reformatting. Use with extreme care! You have to create one Windows partition (holding whatever you have on Windows now) and the rest of the disk; the second partition will be for Linux. Afterward, when the Linux installation procedure is running, you can partition the Linux area of the disk into the boot, root and swap partitions at a minimum to install and configure Linux correctly. When you reformatted the Linux partition, LILO appeared to you as not being erased, because what was left intact after reformatting is the disk's boot sector, which contains Linux's boot loader. To get rid of that, use the Windows (or MS-DOS) FDISK with /MBR as an argument, e.g., **A:>FDISK /MBR**. This will reinstall the normal WIN/DOS boot loader. —Felipe Barousse, fbarousse@piensa.com

The lilo boot code can reside in different places. It can be on your MBR (Master Boot Record), which is the most common configuration, or it can also be in the boot sector of some primary partition if it's flagged as active. To configure lilo, edit /etc/lilo.conf. On my machine, I can boot DOS with

```
other=/dev/sda1
        label=dos
        table=/dev/sda
```

After that, rerun **lilo**. To uninstall lilo, **lilo -u /dev/device** should do, or you can also type **fdisk /mbr** from a DOS boot floppy. In /usr/doc/lilo-0.21/ (or a similar directory), you should have a file called QuickInst. For many more details, it also has a README. —Marc Merlin, marc_bts@valinux.com

Warped Edges

I am new to Linux, so please don't laugh too loud at my question. I am using a 19-inch monitor, with a Stealth IIIs540 Diamond card. I have gotten Linux up, but I seem to be using the wrong screen resolution. It appears kind of bowed out at the edges. How can I change this? —Virgil Denny, freeagain@earthlink.net

If the text and window size look good to you, but just the edges are curvy, chances are you're running at the right resolution, but have some monitor or display attributes to change. Most monitors have controls at the bottom of the unit that allow you to change the width, height, color, angle, and even the convex and concave curves of your display. Try to use this to correct your problem first. If this doesn't resolve your problem, there is a program called **xvidtune** that allows you to adjust these qualities as well. If you do decide to play with xvidtune, make sure to click the auto button first. That will allow you to automatically see the changes you're making when you click various buttons. If you have indeed decided your resolution is not right, one of the most user friendly methods to changing your X (and other) configurations is XF86Setup. Get it installed if you don't have it already, you'll be glad you did. —Kara Pritchard, kara@linux.com

Driving Miss Printer

I need two drivers, one for a HP DeskJet 712C printer and one for a Umax Astra 1220P scanner. I do not understand why they are not supported by my distribution; this one is brand-new. Those drivers should be quite easy to install. I could not find them on any site, had no response from the newsgroups and do not know anybody who can help me. —Marc Nadeau, cerceaux@francimel.com

Something commonly confused by new Linux users is the difference between driver behavior in Linux and other operating systems. For your printer, either run **control-panel** and click on the printer tool, or run **printtool** directly (from Red Hat). You can then choose the driver for the series closest to yours (e.g., HP DeskJet 6xxC series) and your printer should work fine. To configure your scanner, visit http://www.mostang.com/sane/ . That is the web site for SANE, Scanner Access Now Easy. SANE is a universal scanner interface, and their supported scanners page lists multiple Umax Astra scanners that are supported. —Kara Pritchard, kara@linux.com

You've Got Mail

I have set up my Linux machine as a mail server using Sendmail. At this stage, it is just the mail server for the company network. All the other workstations are Windows machines using Outlook Express. I have set up the clients to use the Linux machine as their SMTP and POP3 server. When a user sends mail to another user, the message gets sent fine. Now, instead of the e-mail going to /var/spool/mail/John, for example, the e-mail goes to /var/spool/mail/root. When I read the e-mail in root's mail, it says both the sender and the receiver are named Unknown. I have set up user accounts successfully. How do I get the server to accept mail for each user and to recognize the users? My POP is working, as I am able to use **telnet** successfully with POP, but I cannot get POP-3 to work properly. When I type **telnet localhost pop-3**, it says:

```
Trying 127.0.0.1
Connected to localhost Escape character is ' ^ ] '.
```

then after about five seconds on its own, it says,

```
...Connection closed by foreign host...
```

Where does the Linux server store the user's mail? When a user tries to retrieve their mail, nothing is coming through. How do I configure the server to direct the mail to the user's local machine? —Mark Wainman, wainman@iafrica.com

A user's mail, after it has been received by Sendmail and (most of the time) processed through **procmail**, is stored in /var/spool/mail/userid where “userid” is the login user name. When retrieving mail with POP, mail is read from that location and transferred to the user's home directory, appending the message to the file **mbox**. POP actually reads the mail from mbox in the user's home directory. If your user's PCs are on a LAN, you must properly configure your e-mail clients (Outlook Express) to use SMTP for outgoing mails and POP for incoming ones. The user name and password must be correctly set up for remotely logging in to the Linux machine. Maybe it is worth mentioning that you need to have the POP service installed through the IMAP RPM on a Red Hat system. Also, it has to be enabled; that is, the POP-3 lines should not be

commented out in /etc/services and /etc/inetd.conf. If this is correctly set, you should be able to retrieve mail from the server from the PCs. For Internet e-mail from the PCs through the server, Sendmail has to be correctly configured as well according to your external connection parameters. I think you may have a setup problem with your domains. The Linux box may not be able to find local users due to incorrect configuration of Sendmail, specifically in relation to **localhostname** and **localdomain**. Check the file /etc/sendmail.cw and put in the local domain name, for example “yourcompany.com”, and make sure the Linux machine is named after that, e.g., serverpc.yourcompany.com. —Felipe Barousse, fbarousse@piensa.com

resources

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen M. Dahl

Issue #74, June 2000

FileZerver, AT75C310, Eyelet GUI and more.

FileZerver

Microtest Inc. introduced FileZerver, a network appliance designed to meet the growing file storage and management needs of departments and work groups. Features of FileZerver include Linux-based architecture, scalability, RAID data protection, flexible volume management and cross-platform support. The system's software, embedded in FileZerver itself, is configured and managed through a simple browser-based interface.

Contact: Microtest, Inc., 4747 North 22nd St., Phoenix, AZ 85016-4708, 602-952-6400, 602-952-6401 (fax), sales@microtest.com, http://www.microtest.com/.

AT75C310

Aplio, Inc., in partnership with Atmel Corporation, announced the distribution of the Aplio/TRIO Chip by ATMEL, under the name AT75C310. The AT75C310, featuring an embedded Linux operating system, VoIP and audio-application software together with an application development platform, provides a total system solution, empowering CE manufacturers to launch Internet phones, e-mail phones and MP3 appliances at low cost and short time to market. The VoIP application software delivers true telephone sound quality featuring PacketPlus

Technology. The AT75C310 is a low-cost, single chip that simultaneously handles voice processing, telephony and VoIP protocol stack tasks.

Contact: Aplio Inc., 1670 S. Amphlett Blvd., San Mateo, CA 94402, 800-461-4038, 650-655-4065 (fax), sales@aplio.com, http://www.aplio.com/.

Contact: Atmel Corporation, 800-292-8635, literature@atmel.com, http://www.atmel.com/atmel/products/.

Eyelet GUI

MoJo Designs announced they have ported Eyelet GUI to Linux, making it the smallest embedded GUI available for Linux. Using Eyelet GUI, Linux developers will be able to add a graphical interface without the substantial overhead of the X Window System. Eyelet GUI is particularly well-suited for touch-screen applications such as medical devices, automotive-information systems and a host of consumer products including digital cameras, information appliances, cellular phones, microwave ovens and VCRs.

Contact: MoJo Designs Inc., PO Box 6037, Boulder, CO 80306, 303-443-5035, 303-441-2902 (fax), info@mojodesigns.com, http://www.mojodesigns.com/.

J2SE 1.2.2 for Linux

Sun released the first customer shipment of J2SE 1.2.2 for Linux, an advancement on the Java 2 platform. It enables Linux developers to write and deploy innovative enterprise applications, based on Java technology, to Linux systems. The union of the Java 2 platform with Linux greatly expands the pool of enterprise applications available to Linux developers, enabling Linux end users to take advantage of thousands of Java-technology-based applications.

Contact: Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303, http://www.sun.com/products-n-solutions/.

GNUPro Tools for IA-64

Red Hat, Inc. announced GNUPro tools for IA-64. The compiler and debugger tools for native and embedded software development for IA-64 offer a pre-

production development environment for creating applications on Intel's forthcoming Itanium processor. GNUPro tools for IA-64 include the IA-64 Linux source code, Red Hat's GNUPro compiler and other key software. The GNUPro tools for IA-64 are completely open source and have been released to the open-source development community via the Trillian Project web site.

Contact: Red Hat Software, http://www.redhat.com/, http://www.linuxia64.org/.

Linux edition of "A Mother's Shoah"

AIL NewMedia Publishing announced the release of its new title "A Mother's Shoah" on CD-ROM for Linux computers. The e-book is based on the diary of a Holocaust survivor, Mrs. Susan Kaszas. Archive photographs, images of paintings and pages from the diary provide historical depth and visual impact. AIL NewMedia Publishing also publishes a printed book edition, sold along with the CD by Tree of Knowledge on AIL's web site. Proceeds from this publication will go to the HIAS and the Third Temple Foundation.

Contact: AIL NewMedia Publishing, 31 Franklin Turnpike, Waldwick, NJ 07463, 201-444-5051, inquire@newmediapublishing.com, www.newmediapublishing.com/tok/ams_cd_linux_page.html.

+One Station

Maxspeed Corporation introduced a new Linux desktop product, +One Station. Targeted toward small businesses and home environments, it allows Linux PC users to share software, Internet access, databases and peripherals simultaneously with up to four other users by connecting a monitor and keyboard to the host Linux PC, allowing all to act as a PC. The +One Station currently supports the Red Hat Linux distribution. Support for Turbo Linux, Mandrake, Caldera and SuSE is expected to be announced soon.

Contact: Maxspeed Corporation, 3788 Fabian Way, Palo Alto, CA 94303, 650-856-8818, 650-856-8838 (fax), info@maxspeed.com, http://www.maxspeed.com/.

Parallel Computing Toolkit

Wolfram Research, Inc. announced the Parallel Computing Toolkit, a new Mathematica application package that makes parallel programming affordable to users with access to either a multiprocessor machine or a network of heterogeneous machines without requiring dedicated parallel hardware. The toolkit can take advantage of existing Mathematica kernels on all supported operating systems, connected through TCP/IP, enabling users to create low-cost "virtual parallel computers" using existing hardware and Mathematica licenses.

To create an active connection, the master computer requires Mathematica 3 or higher for UNIX/Linux.

Contact: Wolfram Research, 100 Trade Center Drive, Champaign, IL 61820-7237, 800-965-3726, 217-398-0747 (fax), info@wolfram.com, http://www.wolfram.com/applications/parallel/.

Rave Systems RackMount-1UAXe

Rave Computer Association, Inc. introduced the newest addition to its Rave Signature Series. Rave Systems RackMount-1UAXe (RM-1UAXe) features Sun's newest UltraAXe motherboard integrated into Rave's rackmount 1U form factor chassis. The UltraAXe motherboard is powered by a 300MHz UltraSPARC-IIi CPU with a 256KB external cache and features three PCI slots, an on-board graphics accelerator and on-board EIDE controller that can support up to four EIDE drives. The new system provides a high-performance, low-cost solution operating with a Solaris or Linux environment.

Contact: Rave Computer Association, Inc., 36960 Metro Court, Sterling Heights, MI 48312, 810-939-8230, 810-939-7431 (fax), http://www.rave.net/.

SafeWrite

TurnSafe Technologies, Inc. released SafeWrite, an e-mail security product with advanced modern encryption technology. Key features of SafeWrite include the ability to send secure messages to anyone, without concern for public keys or the receiver's software; use your existing POP, IMAP4, web-based or AOL e-mail account; send messages with dissolving keys, so that after the message is read once it cannot be re-opened; and receive confirmation when your message has been opened. SafeWrite is fully compatible with most operating systems including Linux.

Contact: TurnSafe Technologies, Inc., 1 Arkendo Dr., Oakville, ON L6J 5T8, Canada, 905-681-5753, http://www.turnsafe.com/.

Progress SonicMQ Adds Support for Linux

Progress Software Corporation announced that the Developer Edition of its award-winning Progress SonicMQ Internet messaging server will support Linux.

SonicMQ is based on Java Message Service (JMS), Sun's specification for Java-based messaging. In addition to the Developer Edition available at no charge, SonicMQ is available in an Enterprise Edition and a Small Business Edition. SonicMQ is supported on all platforms running the JavaSoft V1.2 JVM.

Contact: Progress Software Corporation, 14 Oak Park, Bedford, MA 01730, 781-280-4700, 781-280-4095 (fax), sonicmqpurchase@progress.com, http://www.progress.com/sonicmq/.

System Blocks

The Information Technology Solutions Group (ITSG) of SM&A Corp. unveiled System Blocks, an integrated software suite designed to automate system testing, as well as monitor and control distributed application processes in mission-critical environments. System Blocks was originally developed on UNIX and is available in a Linux version. It provides real-time or archival data gathering and analysis, intelligent corrective action on detected problems and automatic control and monitoring of processes.

Contact: SM&A Corp., 4695 MacArthur Court, 8th Floor, Newport Beach, CA 92660, 949-975-1550, 949-975-1624 (fax), www.smaproducts.com/newitsg/itsgweb.

T.Rex

Freemont Avenue Software announced the T.Rex open-source enterprise security suite. This highly integrated firewall combines functions normally requiring installation of multiple products. Application-specific proxies provide high levels of security and access control tailored for the application. The CD-ROM is available in a Standard, Deluxe or Professional edition, and supported platforms include Linux. T.Rex is released under an open-source license. Source code and binaries will be readily available to all users, who will be free to make changes. Any modifications will be shared with FAS and the wider user community.

Contact: Freemont Avenue Software, Inc., 1830 S. Kirkwood, Suite 205, Houston, TX 77077, 281-759-3274, 281-759-8558 (fax), sales@opensourcefirewall.com, http://www.opensourcefirewall.com/.

Videomodem

COM One Services introduced its Videomodem technology, which offers a means of capturing and transmitting real-time or recorded video information to web sites. By installing video cameras at specific areas of interest for information such as traffic or weather conditions, the images are transferred to

an Internet server to be viewed by web surfers in real time. When used as a surveillance system, the Videomodem allows companies to monitor work flow, visitors, or restricted/secure areas easily and confidently. Compact and capable of accommodating up to four cameras, the Videomodem kit includes real-time Linux-based server software.

Contact: COM One Services, 11 parc de Marticot, 33610 Cestas, France, 33-0-5-5797-7272, 33-0-5-5678-8478 (fax), services@com1.fr, http://www.com1-services.com/.

SNA Gateway

Gcom, Inc. introduced a Linux-based SNA gateway for simplifying web-to-mainframe connectivity. It also simplifies the task of connecting an IBM host with remote computers/workstations on a local LAN. The gateway plugs into SDLC, Token Ring or QLLC/X.25 lines to the host and provides TCP/IP access to 3270 and 5250 applications via the TN3270E, TN3270 and TN5250 protocols. Each Gcom SNA gateway system is complete with a stand-alone PC-based server, Gcom SNA and TN server software (installed), synchronous and Ethernet LAN adapter cards, mouse, keyboard, monitor—and a one-year service/support agreement. Bisync support and API for developing custom applications are available.

Contact: Gcom, Inc., 1800 Woodfield Drive, Savoy, IL 61874, 217-351-4241, 217-351-4240 (fax), sales@gcom.com, http://www.gcom.com/.

Best Linux 2000

SOT released Best Linux 2000, the first English version of the #1 Linux distribution in Finland. The boxed set includes a 400-page manual and installation, source code, Linux games and software library CDs, plus lifetime technical support and a free update service. New technical features include extended hardware support for popular Ultra DMA 66 hard disks, multiprocessor support for up to 16 processors, built-in support to VPNs, security-related enhancement software and more. Best Linux is developed and maintained under a GPL license as freely distributable software and is also available on the Internet free of charge.

Contact: SOT Finnish Software Engineering Ltd., sales@sot.com, http://www.bestlinux.net/.

Archive Index Issue Table of Contents

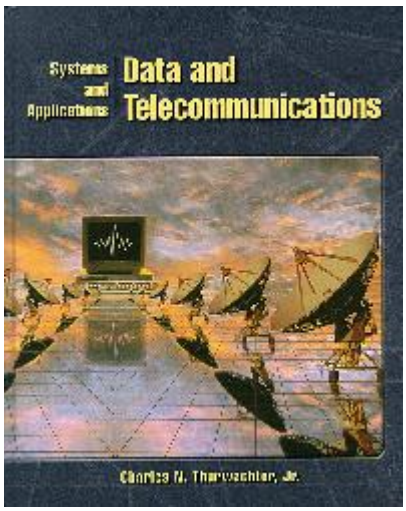
Advanced search

Data and Telecommunications: Systems and Applications

Derek Vadala

Issue #74, June 2000

Billed as an in-depth discussion of data communications presented in a way that is “understandable for students who have a limited mathematical background”, Data and Telecommunications covers concepts behind analog and digital communications and the way they have been applied to telecommunications systems and data networks.



- Author: Charles N. Thurwachter
- Publisher: Prentice Hall
- ISBN: 0-13-793910-8
- URL: http://www.phptr.com/
- Price: $98 US
- Reviewer: Derek Vadala

I have been looking for a book like this for a long time. While TCP/IP, Internet routing and Ethernet are well-covered topics, the telcommunications network that connects the Internet to other data networks, public and private, and to individual organizations and networks is often overlooked.

Many of us use dial-up networking, dedicated circuits (T1, DS3) and DSL on a daily basis, but it's not often that Linux users and system administrators are exposed to the physical or network layers of telecommunications systems. In fact, we're often forced to rely on ISP help desks and specialized phone company personnel who rarely grasp the big picture and are armed with only troubleshooting checklists. This book would make a good reference for any hobbyist or professional who needs to understand these technologies on a more fundamental level, although it might not be the best choice for casual readers.

Billed as an in-depth discussion of data communications presented in a way that is “understandable for students who have a limited mathematical background”, *Data and Telecommunications* covers concepts behind analog and digital communications and the way they have been applied to telecommunications systems and data networks. Thurwachter begins with a concise introduction to the topic, after which he covers electromagnetic signals, metallic (copper) and optical (fiber) media and antennas. These chapters, like most of the book, take an academic approach to their topics. Because of this, I think many readers will find the book a bit inaccessible.

After covering the basics, Thurwachter launches into a fairly academic discussion of signal and fourier analysis, and amplitude, angle and pulse modulation and demodulation. Although the book prides itself on non-mathematical discussions of the topic, I found that mathematical concepts had not been replaced with thorough real-world analogies and clear high-level explanations.

Chapter 11 (multiplexing) will likely have the biggest appeal to readers who come from UNIX and TCP/IP-based networking backgrounds. It covers channelized T1 and DS3 circuits, as well as signaling associated with these circuits. Thurwachter follows with coverage of error correction and digital modulation.

Chapters 14 and 15 attempt to cover the OSI model and networking hardware. These chapters are better covered by Tannenbaum and Stevens and are clearly written for an audience without any networking experience. Chapter 16 on telecommunications thoroughly covers the details of phone-switching equipment and call routing, but it lacks a practical discussion of the topic. For example, in a discussion of 2-wire vs. 4-wire circuits, no mention of T1 being a common 4-wire circuit is made, despite the fact that it is a widely deployed communication link.

The book finishes with a brief discussion of networking protocols, but like earlier chapters, this topic is covered more thoroughly and with better clarity in

other texts. While there is some good information in *Data and Telecommunications*, it's hard to find it in the middle of a book meant for students. Because of its hefty price, this book probably won't make a good addition to home libraries. But since it does contain some useful, hard-to-find information, I'd recommend buying a copy for the office and passing it around.

Derek Vadala lives in Portland, Oregon where he is currently writing a book for O'Reilly. Feel free to contact him at derek@cynicism.com.

Archive Index Issue Table of Contents

Advanced search

Installing Window Maker

Michael J. Hammel

Issue #74, June 2000

Mr. Hammel gives us the basics for installing and configuring Window Maker.

The source for Window Maker can be retrieved from the primary web site at http://windowmaker.org/. You'll need two packages from this site: the source distribution and the **libPropList** source package. The latter package is a library used by Window Maker, which you must build and install before attempting to build Window Maker. End users probably won't refer to it much after building Window Maker, so we'll just look at getting it installed. At the time of this writing, the latest version of libPropList is 0.91 and the latest version of Window Maker is 0.61.

In order to make use of fancy graphics, you'll need the standard set of graphics libraries: libpng, libtiff, libjpeg, libgif and libXPM. All of these are fairly standard on newer Linux distributions. If you have an older version (more than two years old), you may want to check if these libraries are installed. If not, check Freshmeat (http://freshmeat.net/) for where to find the latest versions.

One other optional library is GNU xgettext. This library is needed only if you plan to use a language other then English, and then only if you want the messages displayed by Window Maker to be in that language. You won't need it, for example, if your root menu is in French. Internationalization is a topic all to itself, and since I have enough problems with English, I will leave it to someone better suited to its discussion.

If you're installing from source *and* you use Red Hat, there are a few rules to follow:

1. Uninstall any existing Window Maker installation. Execute these commands:

```
rpm -qa | grep -i window
rpm -e package
```

- package is the package name returned by the first command. If no package name is returned, then skip the second command.

1. Make sure the **LANG** and **LINGUAS** environment variables are not set to **en_RN**:

```
set | grep LANG
```

- If this returns anything, then type:

```
unset LANG
```

- Similarly,

```
set | grep LINGUAS
unset LINGUAS
```

1. Make sure there is a link from /usr/include/X11 to /usr/X11R6/include/X11:

```
ls -ol /usr/include/X11
```

- If this returns “No such file or directory”, then type:

```
ln -s /usr/X11R6/include/X11 /usr/include/X11
```

Building is a breeze if all the prerequisites are installed. I installed under /usr/local/WindowMaker (both the libProplist and Window Maker packages), which required updating the /etc/ld.so.conf file and running **ldconfig** afterwards.

We'll skip installing the graphics libraries, since most users will probably already have these. Once you've downloaded them, unpack the libPropList and Window Maker source packages into their own directories using the following commands:

```
tar xvzf libPropList.tar.gz
tar xvzf WindowMaker-0.61.tar.gz
```

This will create directories called libPropList-0.91 and WindowMaker-0.61. Note that the file name is dependent on whatever the current versions are.

In the libPropList directory, there is an editable configuration file (plconf.h), but this probably isn't necessary. Just run the following commands:

```
 ./configure --prefix=/usr/local/WindowMaker
make
```

The first command configures the source to be installed under /usr/local/WindowMaker. If you've read any of my other articles, you'll know I install new packages in their own directories under /usr/local. Since so many Linux

applications are evolving entities, this sort of product management makes upgrading from source code much simpler. In this case, since I don't expect to use libPropList for anything other than Window Maker, I'll just stuff it into the same place I'm going to install the Window Maker files.

Once the source has been compiled (via the **make** command), you need to change to the root user via **su** or **sudo** and run the following command:

```
make install
```

At this point, libPropList is installed, and you will most likely not have to deal with it any further. Be sure to exit from the root user account.

Building the Window Maker source is just as simple. The configure script has many options. Although you probably won't need to use any of these options, you should still read the INSTALL file to be certain. Options that might be useful include **--enable-kde** to run Window Maker with KDE, **--enable-gnome** to run Window Maker with GNOME, and **--enable-sound** if you like annoying sounds associated with certain actions on the desktop.

I'm not going to be using Window Maker with GNOME or KDE, so my configure command looks like this:

```
./configure --prefix=/usr/local/WindowMaker \
   --with-incs-from=-I/usr/local/WindowMaker \
   --with-libs-from=-L/usr/local/WindowMaker/lib
```

Note: the “\” is a continuation character which allows you to spread a command over multiple lines.

Now, we just need to build and install the package:

```
make
```

Change to the root user and run:

```
make install
```

The last step for installation of the Window Maker source package is to make sure the Window Maker and libPropList libraries can be found when you run the window manager. To do this, run the following commands:

```
echo "/usr/local/WindowMaker/lib" >> \
  /etc/ld.so.conf
ldconfig
```

The first command appends the directory name to the end of the ld.so.conf configuration file. The second command tells the operating system to reload that configuration because a change was made.

The last three commands were all run as root, so now you can exit from the root user back to your normal user account.

The next step is to install the Window Maker data package, which includes a set of pixmaps for use with Window Maker. This package requires manual installation by copying the pixmaps to the proper directory. Change to the WindowMaker-data directory (after unpacking it). If you installed Window Maker in the default directory, /usr/local, then you can use the following command:

```
cp -r pixmaps /usr/local/share
```

In our case, we installed under /usr/local/WindowMaker, so we'd use a command like this instead:

```
cp -r pixmaps /usr/local/WindowMaker/share
```

Finally, make sure the Window Maker binaries and scripts can be found by adding them to your PATH environment variable:

```
export PATH=$PATH:/usr/local/WindowMaker/bin
```

The First Time

Now that you have Window Maker installed, it's time to see what you're getting. Before starting the window manager for the first time, you need to run **wmaker.inst**, a script that should be in the bin directory of the installation (/usr/local/WindowMaker/bin in my case) to set up your user ID to use WindowMaker. This script will check for the presence of “wmaker” in your .xinitrc, .Xsession and .Xclients files using a simple **grep** command. If the script finds such an entry, it assumes WindowMaker is your default window manager. This isn't necessarily the case (my .Xclients, for example, includes a reference to **wmaker** that won't actually launch the window manager), so you may need to configure it manually in order to use wmaker.

There are many ways to configure your login to use a particular window manager or X application at startup. In the previous article in this series, I discussed the use of the .Xclients file. Let's review this process very briefly (if you need a more detailed description, see the previous article on Enlightenment).

All Linux distributions use some script interface to a program called **xinit** to start your X session. These scripts will, if they are the standard scripts that have been used for years, eventually run your .Xclients file to launch applications and your window manager. Therefore, the simplest method of getting Window Maker to be your default window manager is to create a file called .Xclients in

your **$HOME** directory. Chances are this file doesn't exist yet (unless you created it once before). Add a single line to it:

```
wmaker
```

Note that you do not want to place this command in the background. For example, **wmaker&** would be incorrect; this would cause your X session to start up and then immediately exit. You also do not need to specify which shell to use when this script is run. Many scripts begin with something like

```
#!/bin/sh
```

but .Xclients does not need to have such a line.

Once you've created (or modified) your .Xclients file, you're ready to start up Window Maker for the first time. How you do this depends on your distribution, but I log in using an ordinary text console, then type **startx** to get things running. If you use a graphical login, you may need to log out and then log back in.

The first time you start Window Maker, it will create a set of directories for you under $HOME/GNUstep. These directories are where you manage menus and themes. The sidebar describes what you'll find in these directories. Unless otherwise specified, you shouldn't edit these files by hand. Most have graphical interfaces, such as using an application's “Title Bar Attributes” menu option or the “Settings” menu option for a docked applications icon.

GNUstep Directories

If you are running Window Maker using a language other than English, you can switch to language-specific menu files. The INSTALL file in the source distribution describes how to set it up for using alternate languages.

Now you are ready to run. More about Window Maker can be found in my article “Artist's Guide to the Desktop, Part 3” in this month's print magazine.

email: mjhammel@graphics-muse.org

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