Linux Journal Issue #86/June 2001

Columns

Reviews

Departments

Archive Index

Advanced search

Focus: Internationalization and Emerging Markets

Richard Vernon

Issue #86, June 2001

What has surprised me somewhat more is something I've discovered since beginning to work for *Linux Journal*—the impact Linux has had in other corners of the globe, in fact in every corner of the globe.

Having recently returned from CeBIT in Hannover, Germany, where Jon “maddog” Hall of Linux International took excellent advantage of his precious opportunities to explain what Linux offers to key German cabinet members, I have a better understanding of the enthusiasm Western Europe reserves for Linux. Attendees who stopped by the *Linux Journal* booth to express their fervor included Dutch, German, English, French, Spanish, Portuguese and Belgian. But this is to be expected. What Linux enthusiast is unfamiliar with SuSE and Mandrake? Linux is, after all, the offspring of a European mind.

What has surprised me somewhat more is something I've discovered since beginning to work for *Linux Journal*--the impact Linux has had in other corners of the globe, in fact in every corner of the globe. For instance, what I found to be the most impressive Linux product displayed at CeBIT was neither European nor American, but the Mobile Multimedia Communicator offered by Galleo, an Israeli company. Of course I shouldn't have been surprised as Linux started as an international endeavor, a child of the Internet with little regard for geographical boundaries, and cost (except as it relates to internet connectivity) doesn't really constitute a barrier. As I'm sure our readers have noticed, our contributors reside in diverse parts of the world and face diverse computing challenges. Their solutions demonstrate the scalability and flexibity Linux offers.

In these pages Michael Stafford reviews *The Hacker Ethic* in which Linus Torvalds explains the entertainment value of hacking, the pleasure derived from computer creation. Perhaps this pleasure that so easily transcends national boundaries has something to do with the worldwide success of Linux.

Linux, by nature of its flexibility and learning curve, encourages enthusiasm for hacking—something those who employ the more “user-friendly” OSes miss out on.

However, it's not all peace, love and Linux warm fuzzies as Wayne Marshall, in his article “Algorithms in Africa” on page 72 points out. Linux is perhaps best known for powering the Internet, and internet connectivity is behind the hopes to bridge the Digital Divide said to exist between developed nations and those termed Third World. Wayne, a favorite regular contributor who has been living in Guinea for some time, highlights the fact that the divide is less than digital, though still informational—and the Western world may be on the ignorant side.

John Biggs recently spent two years in Poland and in his article “Linux in Poland”, on page 80, discusses to what point Linux has been embraced there and what obstacles remain.

Finally, Jorge Eduardo Nieto Lema reminds us of the potential advantages open-source software presents to nations having more important things to worry about than licensing fees. On page 82 he explains the Linux Terminal Server Project that allows setting up diskless workstations that boot from a Linux network server, a real boon to companies on a tight budget. Jorge tells of his experiences setting up the LTSP for a Colombian company and explains the installation.

—Richard Vernon, Editor in Chief

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Algorithms in Africa

Wayne Marshall

Issue #86, June 2001

Maybe the rush to market for spreading internet access across the globe isn't in anyone's best interest—a report from the front.

Eleven years ago I installed a computer system at a vocational training and development center in Tutume, Botswana. Tutume is a rural village on the northeastern edge of the Kgalagadi desert in southern Africa. The computer was intended to help this organization, known as Tutume Brigades, catch up on its bookkeeping for several business units crucial to the local economy. Businesses included a brick-making unit, carpentry workshop, auto repair garage, sorghum mill, school uniform production unit, tractor hire and vegetable garden. For the local village and the surrounding catchment era, the Brigades were literally the only game in the bush for commodities, trade skills, training and employment opportunities.

When I arrived in Tutume, I was a pure novice in the field of foreign assistance. I was also a mid-career financial professional, with several years of experience in nonprofit and health-care management in the United States. And like most aid workers new on the ground in Africa, I knew what was best. In my assessment of the center, I believed a computer was essential to get a handle on the Brigades' financial position, which otherwise consisted of eight separate sets of badly maintained manual ledgers, over nine months in arrears. Except for the bank statements of eight separate checking accounts (and even the bank statements proved unreliable), we had no way of knowing if the center had any money. Every time we had to make payroll or buy another truckload of cement, we were in the heart of fiscal darkness.

Over the course of the next several months, I proceeded to computerize the records and train local staff in basic operation of the system. By the end of the first year, the financial records of the center were timely and accurate. Moreover, other staff members were beginning to use the computer for tasks

such as word processing and spreadsheets. Many of these employees had never even used a typewriter before.

If I were to tell no more of this story and fade here to one of the glorious Kgalagadi sunsets, this might be called a win. Although set in the predawn (and pre-Linux) history of the Internet era, today this would be described as a small success story of "bridging the digital divide" in Africa—like I was a regular Albert Schweitzer of the Information Age or something.

But the truth is not so simple, and the issues of foreign assistance are not so trivial. The fact is, I am not proud of this story. Because as my time in Tutume went on, I realized I had blundered badly, to the point of putting the Brigades in serious jeopardy. I began to ask myself such basic questions as: What would happen to the computer after I left? Was the staff fully capable of operating the system independently? Would backups be maintained and performed rigorously? Were skills sufficient to troubleshoot problems and reinstall the system if necessary? If the equipment failed or was stolen, could the center afford to replace it? And what would the center do when the staff I had trained for so long were lured away by more lucrative jobs in the big city?

These questions all led to the same answer: the Brigades would be left in even worse shape than I found them. Rather than gaining empowerment, independence and enablement, they would more than likely be left powerless, dependent and possibly ruined. And all because of my own cultural myopia, despite my good intentions.

It is axiomatic in the field of foreign assistance that the aid program will take credit for the successes, while failures are blamed on the host country. The psychology of failure can then be even more severe and long-lasting than the loss of the project. While I was working in Tutume, for example, a friend of mine was working in the village of Lobatse in southern Botswana. Seven years earlier, an aid organization from northern Europe had decided a wool sweater factory would be just the ticket for the economic development of the village. Of course, northern Europeans are fond of nice wool sweaters and very likely have great need for them, particularly in the colder climes of northern Europe. The market for wool sweaters is less extensive in the sweltering and sparsely populated Kgalagadi desert, however. After seven years of subsidizing the losses of the operation, the aid organization finally decided it was never going to be sustainable, and they pulled the plug on the effort. My friend's unenviable assignment was to put all the women out of work, sell the facility and liquidate the equipment. It was hard for many of the women not to feel that the fault was somehow their own.

Fortunately for Brigades in Tutume, such failure was averted. As the story there continues, once I realized the risks, I spent the next several months converting the accounting system back to manual ledgers, hiring and training additional staff in bookkeeping procedures and enabling them to use the computer primarily as a support system, rather than as the central financial database.

But what do these stories from Tutume and Lobatse have to do with Linux and emerging markets? The rest of this article will consider that question.

The Digital Divide

Nine years have passed since I left Botswana. To put the times into perspective, the first thing I bought when I got back to the US was a fax modem, the cheapest, fastest solution to stay connected with the contacts I had made abroad. My modem then was 2,400 baud. I tried out CompuServe and decided on Delphi, and the buzz was just starting about something called PPP.

During the next several years I was in and out of Africa, became a Linux user in 1995, began installing Linux in nonprofit organizations in 1997, spent a year and Y2K transition in the former soviet state of Ukraine and came to the West African country of Guinea in May 2000. At some point during this period the digital divide was invented.

Actually, the digital divide seems to have its origins in a 1995 report from the US Department of Commerce, whose National Telecommunications and Information Administration (NITA) released the first paper in a series titled "Falling through the Net". This report analyzed telecommunication access by geographic and demographic patterns throughout the United States. One of the conclusions of the report was the gap between the "information rich" and the "information poor" had widened.

In the later years of the Clinton administration, the digital divide broadened beyond US borders to encompass the globe. The issue gained considerable publicity after a G8 economic summit meeting in 1999, where the most powerful nations on earth decided that the growing gap in information technology was one of the most serious problems facing development in the Third World.

Now, as I write this, bridging the digital divide has become one of the hottest trends in foreign assistance, and many aid organizations and corporate philanthropists have found publicity for their efforts. Simplistically, it seems, the gap in information technology has now come to be identified with access to the Internet. Thus, we have such programs as the USAID-funded Leland Initiative, designed to bring internet access to Africa; the Peace Corps announcing an information technology initiative in partnership with AOL; and a recently

formed organization called Geekcorps sending its second group of volunteers on three-month stints designing web sites in Accra, the capital of Ghana in West Africa [see *LJ* April 2001 for more on the Geekcorps]. Naturally, the high-profile publicity given this issue has created an opportunity for many international aid organizations to develop projects and funding appeals for serving the digitally needy.

The New Tech Testament

Delivering the miracle of the Internet is the new zeal of the high-tech missionary. In what seems to be a rush to market—bringing the Internet to the developing world—sometimes projects are announced with only naïve regard to the technical issues and without full consideration of whether such projects are viable, appropriate, relevant and sustainable. Thus, one hears of a women's cooperative in Central America marketing their handcrafts over the Web; advocates describe the potential of "telemedicine" for delivering virtual health care to isolated areas; and the US State Department Global Technology Corps proclaims, "We have seen farmers in Mexico using [the Internet] to check weather conditions and crop prices."

Where once Norwegians may have seen wool sweaters, the tech visionary now sees web browsers.

At the extreme, the new economy proselyte promotes the Internet as the solution for everything from education and health care to pollution, inequality and world peace. As though everyone who has access will be able to browse their way to nirvana, as though the path to heaven is paved with bandwidth. The satellite dish is the new icon of the digital evangelist, replacing the holy cross.

One of the implicit beliefs of this testament is that information, in and of itself, is sufficient to promote economy, remedy problems and narrow inequities. A corollary implication, the message from one side of the divide to the other, is that we have information and you don't, that our information is good and yours is useless. This is the lesson CNN preaches to its international audience when it tells us, "The human without information is nothing."

It should be clear that in this form, divide rhetoric is simply new raiment for the familiar old taxonomies of prejudice that have long sought to divide the world between believers and heathens, the enlightened and the savage. From a historical perspective, rather than helping, these kinds of belief systems have generally been devastating to their targets.

More importantly, the belief in the sufficiency of information and information technology is simply wrong. Information alone doesn't help people. If only this

were true, doctors would be made from medical textbooks and entrepreneurs would be born from accounting manuals.

In fact, the developing world is littered with unused X-ray equipment, broken-down tractors and empty schoolrooms contributed over the years by well-intentioned and simpleminded donors. These resources are made useless not from missing user manuals or lack of web access, but by the lack of trained technicians, mechanics and teachers.

In short, what empowers people are skills.

Even in the US, this kind of awareness is emerging. In “How Does the Empty Glass Fill? A Modern Philosophy of the Digital Divide” (*Educause Review*, Nov/Dec 2000), Solveig Singleton and Lucas Mast write: “From the standpoint of higher education, students who leave high school without exposure to digital learning tools such as the Internet will prove a much less serious problem than students who leave high school with inadequate reading or math skills.”

And the leading journal of free-market capitalism, the *Economist*, recently observed:

> The poor are not shunning the Internet because they cannot afford it: the problem is that they lack the skills to exploit it effectively. So it is difficult to see how connecting the poor to the Internet will improve their finances. It would make more sense to aim for universal literacy than universal Internet access.

It may be that, with the recent outbreak of dot-com bankruptcy and declines in the stock market, the tenets of the digital religion could be losing their currency. At a time when the mega-billion, IPO-funded ebiz stars like Amazon and Yahoo are having a tough go across the US and Europe, it's hard not to wonder how the promises of e-commerce could possibly prove viable and sustainable elsewhere, particularly in places where there aren't even good banking and credit systems. And for someone like me who has lived several years of the past decade in both rural and urban parts of the developing world—where most of the population still cook with firewood and carry water in buckets—the practical value of focusing foreign assistance on IT projects would seem negligible, if not ludicrous entirely. Given the more serious fundamental issues facing developing nations—health care (AIDS, TB and malaria), nutrition, sanitation, education, poverty, pollution and political corruption—providing the means to surf the Web should probably fall fairly low on any reasonable scale of human priorities.

So is there any way to make a difference, a real difference that improves people's lives? Is there any role for Linux and open-source advocacy in

emerging markets? Are there ways of using technology for solving human problems in places like Africa, without trying to sell wool sweaters in the desert? I wouldn't be writing this article if there weren't.

Algorithms in Africa

When it comes to Africa, the so-called digital divide is just a divide; there isn't anything especially digital about it. The divide is geographic, because Africa is a long way away, and cultural, because the traditions and histories of Africans developed independently from those of Europeans and Americans. Almost incidentally the divide is economic, from the standpoint of cash resources and differing perceptions of wealth, though the natural resources of this continent are vast. The divide ends up being mostly one of ignorance, and this gap is at its widest in America.

Americans in general know very little about Africa, and what little they do know or think they know is usually prejudiced and fallacious. If I were to know the state of Florida only from news reports, I would think it was a large mobile-home park of fat pink people constantly flattened by hurricanes. Similarly, most Americans probably only know Africa as a disaster zone of epidemic, starvation and genocide. The principal media image Americans hold of African assistance is usually the one of the brave young (white) woman, a nurse or volunteer, holding a helpless black infant, center stage among a group of grateful and admiring Africans in the background.

Of course Africa is nothing like this image at all, and the first step in crossing the divide here is to banish these offensive stereotypes and learn all one can about what Africa is really like. It would be a disservice to the many peoples of the continent to generalize and describe the essence of Africa as though it were a single place. But I would just like to say: Africa is such a joy! Whenever I am in the streets of Conakry or an upcountry village, I am overwhelmed with the pure bandwidth of humanity, of color and vitality and life. So much more than can ever be expressed on even your largest CRT, with even the fastest DSL connection; Africa is the ultimate realization of broadband in culture and diversity, natural and human content. Maybe a virtual, flat-screened reality over the Internet is meaningful in the pitifully dreary cubicle of the US office worker, but Africa is all about face time in real time.

Open-source advocates can be sure that Africans get community; Africans get bazaar. These are concepts intrinsic to the cultures and traditions throughout the continent, where African societies had mastered networking long before the invention of the RJ45 jack. Africans have historically been quite receptive, often at their ultimate peril, to ideas and innovations flowing between cultures and brought in by outsiders. And in general Africa has been early and enthusiastic about adopting new communication technologies, particularly

when they are practical and affordable. So in Botswana I was astonished at the number of fax machines per capita ten years ago, and now find a thriving trade in cell phones, both legitimate and black market, in Guinea. On a recent visit to a mosque in the interior of the country, a wizened old muezzin took me up into the minaret specifically to show me their solar-powered amplifier and loudspeaker system, used to call the village to prayers.

As one learns to develop an appreciation of what Africa is really like, it will then help if one can develop a sensitivity to the pitfalls of foreign aid and the havoc such programs have brought to this continent. The subject of other narrations, it is sufficient to observe here that the official assistance programs of foreign governments are usually a foul brew of political hegemony, economic imperialism, cultural ethnocentrism, personal avarice and, too rarely, genuine altruism. Too often the implementation of foreign aid is all about developing market share and spheres of influence, instead of improving lives. Proponents of foreign assistance may even argue that these are synonymous, as though markets for American soft drinks, snack foods and beauty products result in happiness and prosperity for the consumer. The sad fact is, whether intentional or merely consequential, foreign assistance has often had devastating effects on communities, local markets, traditional cultures and environmental conditions throughout Africa.

Finally, it is helpful to bring an honest perspective of one's own history and culture into focus. For example, the United States represents less than 6% of the world's total population and has existed for less than a blink of an eye in the span of all human history. So, what makes us think we've got it right? What evidence is there to suggest this brief record is proof that our way of life and cultural adaptations will be viable in the long run?

For example, it may be surprising to learn that, due to the predations of infectious illness, urban population levels were not even sustainable until about 100 years ago and required steady migration from rural areas. And it was less than 90 years ago, Gina Kolata writes in *Flu*, when "*Ladies Home Journal* proudly declared that the parlor, where the dead had been laid out for viewing, was now to be called the living room, a room for the living, not the dead."

Shortly after this proclamation, a global flu of epidemic proportion—the origin of which is still not understood—killed 1.5 million Americans and 40 million worldwide. This was not in the murky history of the Dark Ages; this was 1918. Today, with the modern plague of HIV/AIDS, the re-emergence of tuberculosis and new mysteries like the relationship of human CJD to Mad Cow Disease, will our mastery of medicine prove all that enduring, even for the world's most fortunate few?

In any case, those who would help others should at least try to learn from past failures and have the humility to ask if the modern model of urbanization, congestion, resource utilization and environmental depletion are sustainable, even desirable, let alone worthy of export to others in the world.

Then we may be able to accept that the Internet may not be the solution to all problems of humankind and have the patience to realize that working through the major challenges in Africa will take time and understanding measured in generations. Now it becomes clear that Linux and open-source developers are helping Africa best by what they have been doing already. People who are programming and installing the world-class, free software at the soul of internet technology are helping others around the world in profound and important ways, no matter what license they are using. GNU and open-source software are the perfect fit for the emerging nations of Africa—as for the rest of the world—not only for the superior technical quality of these systems, but for the values embodied in their development.

The mere existence of Linux and open-source systems give people the chance to use these powerful technologies for low-cost, grassroots level applications, an opportunity not possible just ten years ago. The pages of this magazine have described many of these self-directed success stories, everywhere from Mexico to Pakistan, where Linux solutions enabled people to make the difference. Such examples are to be found among African communities as well, from South Africa to Kenya to Nigeria. And Africans like Katim Touray are using Linux servers to connect other Africans in dialogue around the world.

Beyond the software itself, though, it is the culture of Linux and Open Source communities that provides the model for meaningful outcomes. This is the culture of sharing and empowerment, of the thousands of Linux users' groups throughout the world, of the Linux Documentation Project and the general willingness of one user to selflessly help another. Participating as a Linux user is all about developing crucial skills and passing them on. Often users' groups hold regular installation clinics, giving new users personal, one-on-one support from an enthusiastic peer. And these users' groups are often active in other community projects, such as helping schools install servers and network connectivity, while transferring the skills necessary to maintain them. Each of these connections is essentially more human than technical, linking people together more than their machines, and can lead anywhere. Each of these personal connections sows the seeds of others, and the spread of the Linux bloom is now reaching to every corner of the earth. For example, even though the use of internet technology in Guinea is nascent, Linux certainly preceded my own arrival here. One finds Linux books in French in bookstores and Guineans eager to learn more about this “true” operating system.

And there are other instances of Linux and open source helping to solve problems in Africa. One of the most inspiring and hopeful to me involves no computers at all.

Vim in Uganda

The emergence and spread of AIDS has been devastating to sub-Saharan Africa. Sure, you are probably tired of hearing about it. For one thing, it is so hard to come to grips with the scale of the problem. In the short time since I left Botswana—when AIDS was just beginning to emerge as an issue there—life expectancy has plummeted, from nearly 60 years to barely 40. It is now estimated that as many as 40% of the adults in Zimbabwe are HIV positive. This has been a debilitating setback to the emerging countries of the region, where public health efforts had previously been making remarkable gains.

The epicenter of AIDS in Africa has been Uganda, which was hit first and perhaps hardest. The government of Uganda is considered to have mounted an effective and ongoing public health campaign for its people, and there is hope that the incidence of HIV/AIDS is decreasing. Nevertheless, the consequences of the disease have been severe. One of the biggest problems is the large numbers of children left without parents. In a society where children are traditionally treasured and raised with the supportive assistance of extended families, there are simply too few adults left to care for growing numbers of orphans.

Bram Moolenaar is the author of Vim, one of the most popular open-source text editors, with ports available for just about any platform in existence. Bram had already started Vim when he first went to Uganda in 1994, volunteering to work as a water and sanitation engineer for the Kibaale Children's Centre (KCC).

The center, located in a rural village of southern Uganda, provides food, medical care and education to about 600 children, most of whom have been orphaned by AIDS. The conditions are austere: one book for ten children, a tiny blackboard and a roof with holes.

Bram found that his skills could help at Kibaale, his help made a difference. After a year spent working with the Centre, he wanted to find ways he could continue helping the project while also letting other people know of its existence.

That's when Bram hit on the idea of “charityware” for Vim. The license for Vim says simply: “Vim is Charityware. You can use and copy it as much as you like, but you are encouraged to make a donation to orphans in Uganda. Please read the file doc/uganda.txt for details.”

While using Vim, type **:help uganda** to get the complete text of the license and a description of the Kibaale Children's Centre.

Beyond this, though, Bram is fairly modest about the project. Although he asks for copies of CD distributions that include Vim, he doesn't appeal to distribution vendors directly for any additional financial support. Bram prefers to remain low key rather than risk annoying people and turning them away from supporting the Uganda project.

Knowing that Linux distributions in use are now in the billions, one may wonder how successful the charityware license has been as a fund-raising method for the Centre. Vim users are asked to make contributions to the International Child Care Fund that Bram and his colleagues have set up specifically to support the KCC project, and the ICCF web site provides annual financial reports. For 1999, donation income totaled about $7,000 US (17,800 Dutch Guilders), up from about $3,500 US in 1998.

These figures may seem rather underwhelming and suggest that the conscience of open-source users and vendors is not as evolved as one may like to think. But the bottom line for Bram is, even at such a modest level, these contributions make a huge difference in what the KCC can accomplish. The funds raised by Vim donors are used to keep the Centre running, maintain and improve the facilities and recently purchased rainwater tanks so that more people have access to clean water.

Bram continues his personal involvement with Kibaale to this day, having made return trips in 1996, 1998 and 2000. This experience gives Bram a thorough grounding in the realities of life in Africa, as well as an understanding of the means of effecting meaningful change. When I asked for his opinions about the digital divide, he said, “I'm afraid I don't know what the digital divide is. Is it about bringing computer-related stuff to Third World countries? Well, the area around Kibaale first needs a good water supply and a phone.”

When asked if he could give any suggestions to those interested in projects supportive of African information technology, Bram replied, “The best suggestion I can make is to work in small groups. A hundred small projects bring more benefit than one project that's a hundred times bigger. The strategy and planning done by people in head offices is a waste of time and money.” The message here is that the strength of any bridge depends upon its integrity.

In the end, Bram is doing what the Open Source movement has been all about from the beginning: working with personal conviction, making a difference where one can and sharing the work one loves with others. These are the ideals of a world seeking connections, the values that can link Linux and the Internet

with an orphanage in Uganda. The human connections of these efforts empower people, improve lives and build the solid bridges of understanding among diverse global communities, digital and otherwise.

Resources

Wayne Marshall (guinix@yahoo.com) is a UNIX programmer and technical consultant living in Guinea, West Africa.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux Terminal Server Project

Jorge Eduardo Lema

Issue #86, June 2001

Reduce costs and improve productivity with easy-to-install diskless workstations.

The Open Source movement can help many people and institutions in Latin America by giving them access to technologies and knowledge that are otherwise restricted to developed countries. The lack of economic resources, proper infrastructure and technical education for the vast majority of the population are the main reasons why Latin America is far behind the first world in the use of computer technology, and they have prevented it from reaping the benefits of the information revolution the Internet has created. For example, less than 10% of the population of Colombia (estimated at 40 million) has ever logged on the Internet. This is not an exception, and most countries in the region suffer from similar problems. By using cost-effective, easy-to-implement, open-source technology and free software, these countries can catch up with the industrialized world.

There is an open-source project that could make a huge difference in Latin America: the Linux Terminal Server Project or LTSP (http://www.ltsp.org/). The appeal of this project is that it provides an easy way to set up low-cost, diskless workstations that boot from a network server running Linux. Today, many public and private Latin American institutions can hardly afford new computers or expensive software license fees. By using diskless workstations and Linux, they can save large amounts of money and become much more productive. This technology may also be used to teach children and workers the technical skills they require for the future.

In this article I will discuss the LTSP, describe my experiences with it in a small institution in Colombia and examine what is needed to implement and configure it. This article is merely informative, and it's not intended to be an LTSP HOWTO. Precise installation and configuration instructions can be found at the project's web site (http://www.ltsp.org/documentation/).

What Is the Linux Terminal Server Project (LTSP)?

The goal of LTSP is to create a simplified way of setting up diskless workstations for Linux. By definition, a diskless workstation is a computer that boots after downloading its operating system from a server on the local area network. The LTSP tools accomplish this process by adding small kernel images, XFree86 servers and several other network tools to the server that will pass them to the workstation on request. The diskless workstation only needs a boot ROM for its specific network card in order to obtain the necessary software from the server. Making good use of the open-source software from the Etherboot Project (http://etherboot.sourceforge.net/), the LTSP has created its own boot ROMs and employs them with its software. Understanding that building (burning) a boot ROM for a network card may be a difficult task, a company close to the project, Diskless Workstations (http://www.DisklessWorkstations.com/), offers preconfigured ones for a small price at their web site. It's worth noticing that LTSP's boot ROMs can also be written to a floppy disk from which the diskless workstation can boot.

Our Experiences with the LTSP

I started implementing diskless workstations based on the LTSP tools last year, when a small association of sugar cane industry workers needed a low-cost, easy-to-use computer and information management solution for their headquarters. The company, Productivos Ltda., had a very tight budget, and its office workers were not particularly familiar with computers and software. They had only one computer running Windows 95 with an unlicensed version of MS Office to handle letters, accounting and payroll. Most of the office work was still carried out manually. The new manager of the association wanted to buy five more computers to cope with all the work more efficiently. She also wanted a local area network so people could share files and printers and have access to the Internet. While looking on the Net for a low-cost solution to their problem, I came across the LTSP. The solution was at hand. Productivos Ltda. did not have to buy expensive, state-of-the-art computers or pay for costly software license fees. They were going to have an LTSP network and use StarOffice as their productivity tool.

A test of the LTSP software was conducted before the actual installation, using an old diskless Pentium computer and booting from a floppy disk. After a few tweaks that included the correct setting of the international keymaps under X, I decided this was the perfect solution for Productivos Ltda.

The association bought four new AMD K6 class diskless computers with 32MB RAM each (the lowest configuration available) to act as diskless workstations and a PII 350MHz with 128MB RAM to act as server/workstation. Even though this looks like a normal configuration for many small offices, Productivos'

budget would have normally been able to afford fewer computers under other circumstances (i.e., a Windows network). Just in software license fees, Productivos Ltda. saved around $3,000 US. To avoid booting from floppy disks, the workstations needed bootable network cards. Four Linksys 10/100Mb bootable network cards were bought directly from DisklessWorkstations at $34 US each. A matching 8-port Linksys 10/100Mb hub was acquired from Outpost.com (http://outpost.com/). The required UTP Level 5 cabling was ready in less than a week. The hub and cabling both cost around $350 US.

Red Hat 6.2 (http://www.redhat.com/) was the distribution of choice for the server because it had been fully tested, and it's easy to configure and maintain. Since all the diskless workstations were going to run X and StarOffice 5.2 (http://www.sun.com/staroffice/) remotely, we decided to stay away from memory-intensive GUIs like GNOME or KDE that would reduce the server's performance. We chose to use a window manager with the basic functions only (start bar and menus) and selected IceWM (http://sourceforge.net/projects/icewm/), the lightest window manager around. Only a couple of entries from IceWM's main configuration files (menu, winoptions) were added or deleted to fit the configuration we wanted. Ten different user IDs were created on the server, each with its own StarOffice workstation installation. It's worth noting that StarOffice is a memory-intensive application, another reason why we chose a light window manager instead of GNOME or KDE. The installation of printers was quite simple. A replacement for Red Hat's print tool that adds an LTSP printer to the options is part of the project's software. Now the new LTSP network was ready. The entire installation and configuration was completed in just one day.

The test by fire came when the users logged on the network. They were enthusiastic about the fact that they now had computers to help them with their work. Nevertheless, they needed some training before actually using the computers. All the users went through a two-week in-house training course to familiarize themselves with the new tools available. The training included computer basics and the uses of StarOffice. At the end of the course they were all working without problems and using their diskless workstations for their (previously manual) tasks.

The association's LTSP network has been running smoothly for more than a year now, and they are even planning to add some more diskless workstations. Productivos Ltda. became much more efficient and, at the same time, gave its workers the chance to learn and explore new technologies. All of this would have been difficult without the help of the LTSP and the Open Source movement.

What You Need to Make It Work

System requirements for an LTSP network vary depending on the number of concurrent diskless workstations accessing the server and the applications they will be running. The two most important things to consider here are the server's available memory and the speed of the network. Even though a 10Mb network can handle a large number of diskless workstations running console-based applications, a faster (100Mb) network is required if the workstations are to run remote X sessions and productivity applications like StarOffice. The minimum server configuration for an LTSP network is a Pentium-class computer with at least 64MB RAM and a 2GB hard drive. On the workstation side, a 486 or K5 computer with 16MB RAM and a 1MB video card (for X) can get the work done.

The whole LTSP network operation depends on the good condition of network cards, hubs and cabling to run properly and smoothly. Please be sure to check them prior to any installation.

The Software

LTSP tools currently run on the following Linux distributions:

- Red Hat 6.0, 6.1, 6.2 and 7.0
- Mandrake 7.2
- SuSE 6.2, 6.3, 6.4, 7.1 and 7.2-beta
- Debian 2.0, 2.1 and 2.2
- Caldera eDesktop 2.3, 2.4 and eServer 2.3

All the software and documentation needed to run an LTSP network can be obtained from the download section of their web site, http://www.ltsp.org/. The necessary precompiled packages come in both RPM and TGZ formats. Remember to read the documentation before attempting to install the software and scripts.

Quick Installation Guide (Extracted from the LTSP Installation Guide)

The Quick Installation requires that your installation meets the following criteria:

- You are installing on a Red Hat 6.0, 6.1, 6.2 or 7.0-based system or a Mandrake 7.2-based system.
- The server will have an IP address of 192.168.0.254.
- The workstations will have addresses in the range of 192.168.0.1 through 192.168.0.253.

- Applications will run on the server, displaying output on the workstations.
- You are using DHCP.

Download the LTSP RPM packages from the LTSP download site, http://www.ltsp.org/.

- lts_core-2.xx-xx.i386.rpm—the Core LTSP package, which contains the root filesystem, configuration utilities and documentation for the workstations.
- lts_kernel_xxxx-2.xx-xx.i386.rpm—precompiled kernels for diskless booting. Choose the appropriate kernel for the network card of the diskless workstation.
- lts_xxxx-2.xx-xx.i386.rpm—precompiled X servers. Choose the appropriate X server for the video card of the diskless workstation.

Install the rpm -i lts_core-2.xx-xx.i386.rpm, rpm -i lts_kernel_xxxx-2.xx-xx.i386.rpm and rpm -i lts_xxxx-2.xx-xx.i386.rpm packages.

Verify that **dhcpd** is installed on the server. Run the following command:

```
rpm -qa | grep dhcp
```

It should report a line like:

```
dhcp-2.0-5
```

If it doesn't, then you need to load the DHCP RPM from the Red Hat installation CD.

Once the installation of the above packages is complete, you need to move to the /tftpboot/lts/templates directory. Several files there will configure the system files on your server. Each one of these files is responsible for one system file. Take a look at those files, and make sure you agree with what they are going to do. They can potentially make your system vulnerable to intrusion. You may wish to make changes to the system files manually. If you want to do it automatically, then run the **ltsp_initialize** command:

```
cd /tftpboot/lts/templates
./ltsp_initialize
```

Copy the /etc/dhcpd.conf.example file to /etc/dhcpd.conf. Modify the dhcpd.conf file to include the MAC address of the network card in the workstation. Then, add the following line to the /etc/hosts file:

```
192.168.0.1    ws001
```

You should next edit the /tftpboot/lts/ltsroot/etc/lts.conf file to make sure the entries are correct for the workstation. Then reboot the server and turn on the workstation. You should get a graphical login prompt on the workstation. You can log in using any user ID available on the server.

As you can see, installation of an LTSP network is straightforward, and it can be up and running in a few minutes. Don't forget to follow the instructions carefully, and if you have any difficulties, read the troubleshooting section of the documentation. The project's mailing list is also a good source for solving problems, finding out about new software developments and learning from other users' experiences.

Conclusion

Some important features elevate the stature of this open-source project: its up-to-date documentation, which guides the user through the entire installation; configuration and troubleshooting process; the project's contributions area where people have added their own enhancements to the software, including support for things such as LDAP and dynamic DNS; and its very active mailing list, where most of the questions are swiftly answered by the developers themselves.

The LTSP is now part of SourceForge—good news for users and developers alike. Many new contributions and enhancements to the software are sure to come in the near future, making LTSP the tool of choice for many diskless workstation networks in Latin America and the rest of the world. The recent appearance of a Red Hat-based distribution by the K12 Project (www.riverdale.k12.or.us/linux/k12ltsp.html) includes software from the LTSP. That the K12 Project is aimed mainly at schools and children is proof of the great advancement and significance of the LTSP.

The project's documentation has been translated into Spanish but not yet to other languages. We hope that the LTSP's affiliation with SourceForge means many eager translators from different countries will soon start contributing their work.

Since my first experience with the LTSP, I have had the chance to implement its diskless workstations solution in several environments, including public schools, internet cafés and small companies. To our satisfaction, all of those networks are working smoothly today and most of the users have benefited from the new technology. Maybe this is what the Open Source movement is all about: helping people help themselves.

I'd like to thank Jim McQuillan (jam@McQuil.com) and his friends for their outstanding contribution to the Open Source movement, the development and maintenance of the Linux Terminal Server Project.

Jorge Eduardo Nieto Lema is a 32-year-old Colombian who works as an independent Linux consultant. He is currently working on a port of the LTSP's software to his favorite Linux distribution: the great Slackware. He can be reached at jnieto@yupimail.com.

Archive Index Issue Table of Contents

Advanced search

Polish Linux

John Biggs

Issue #86, June 2001

Linux found quick acceptance and a knowledgeable developer audience in Poland.

Five minutes from the center of Warsaw and the city's looming. A Soviet-era palace of culture, it's a place where software is almost free. As in many Central and Eastern European countries, software piracy is rampant, and you can buy a copy of Microsoft Windows Millennium Edition or a Ricky Martin compact disc for about five dollars at the Stadion.

The Stadion is a crumbling stadium now home to a weekly bazaar featuring pirated pop music and games, cheap clothes and Russian memorabilia like rusting bayonets and aviator goggles. Although recent legislation passed by the Polish parliament has reduced piracy over the past three years, a 2001 study by the International Intellectual Property Alliance estimated that 85% of entertainment software and 55% of Windows business software purchased in Poland is pirated.

The market the pirates haven't been able to crack is Linux. The major technical journal in Warsaw, *Linux Plus*, publishes entire Red Hat distributions for about four dollars, and the Polish Linux Distribution (PLD, http://www.pld.org.pl/) is gathering steam as students and professionals adopt the free OS. Although copies of Red Hat sometimes show up at the Stadion, they disappear in a few days; they just don't sell, so the pirates use the CD jewel boxes for more expensive software.

“A lot of young people are realizing Windows just isn't it”, said Tomasz Kloczko, spokesperson for PLD and a programmer at the Polytechnic University in Gdansk. “There's a wave of new users coming out of the universities and middle schools, and you're going to see Linux become accepted commercially over the next few years.”

In Warsaw's city center, at the Hotel Jan Sobieski III, another free software revolution is taking place. The entire hotel, except for a pay-per-view TV system and a copy of a well-known graphics program, runs on the SuSE and Mandrake distributions of Linux. Everything from the room locks to the hospitality systems are all running on open-source software. The company that installed the software, Pro-Test, and consultants from the Warsaw-based company Softomat translated StarOffice from Sun Microsystems into Polish. The project not only saved the hotel thousands of dollars, it released a new localization of the StarOffice suite to the Open Source community in the process.

“There's one rule that a lot of people have realized: if you need a solution for something, it's probably already been done under Linux. That's the incredible thing about the platform”, said Jaroslaw Szumski, editor of *Linux Plus*. “Businesses are moving over to Linux because people are looking for practical solutions to big problems.”

Szumski said that with the recent crackdowns on piracy, people are looking for other basic software for the office. “The standard in businesses here is of course Windows NT and Office 98, but there is a real trend of people looking at StarOffice or another office package and saying `Why don't we use that?”'

But the universities and internet service providers are a real bastion of Linux in Poland. PLD, a distribution based on Red Hat Linux and translated and localized for Eastern Europe, was a product of necessity. Students and professionals in Warsaw, Wroclaw and Krakow, cities famous for their technical schools, found that most distributions lacked an easy interface in Polish.

“We started with a Polish install program. But necessity is the mother of invention: when we needed something for the distribution, we put it in”, said Kloczko.

The PLD group's work has spread to the Linux Documentation Project. They're now translating technical documentation into Polish and updating HOWTO documents.

With the introduction of the Internet in Schools program led by the Polish Department of Education, PLD took off. Polish versions of applications are showing up in schools around the country.

“PLD is stimulating a change in computing. It's clear that there are a lot of people working on this, and it's a self-perpetuating project. The more you do, the more there is to do”, said Kloczko.

Linux is also coming into its own commercially. One company making waves in the open-source world is ABA (http://www.aba.krakow.pl/), based in the royal city of Krakow. Its flagship product, ABA-X, is a general-purpose thin network client based on Red Hat embedded Linux and running on a Flash-ROM system for easy upgrades.

ABA's owner, Tomasz Barbaszewski, is proud of his recent work with the Polish Parliament. His company installed twenty workstations there and has a contract with Polish Customs for 200 ABA-X stations, complete with passport readers.

“Open source is the only way to create something like this in a short time. To create the code for one of these workstations from scratch would take a few years”, he said. “We didn't choose Windows CE for our operating system because you don't know what's going on inside the OS, and you can't find out. I'd rather be able to create my own system than depend on a mass-produced installation where the bottom line is most important, not stability.”

Barbaszewski sees a vacuum in the Open Source community when it comes to good end-user applications.

“Linux comes in through the back door, in a server. But you still don't have applications. You see one or two solutions for certain clients, but those are closed projects and you don't see code reuse”, he said. “Say you own a business, and you want to use Linux. You have Linux, StarOffice and not much else. You need an accounting program, a money manager—real applications.”

Everyone in the Polish Open Source scene echoed Barbaszewski's complaints. “Everything is in place”, said Kloczko, “but nobody wants to do the boring stuff that will make Linux easier for end users.”

Years ago, as Poland stumbled out of years of Soviet rule, computers like the Atari 800XL and the TRS-80 cost as much as a small car, and mainframe accounts were coveted among university professors. Now, the country boasts four major mobile telephone operators (running Linux), countless ISPs (running Linux) and two major parallel computing test beds in Gdansk and Czestochowa (of course, running Linux). It's one of the fastest-growing economies in the area and business is booming in the Linux consulting market.

In the Middle Ages, Poland created its first national network by building “eagle's nest” castles along a north-south parallel in order to protect the country from invaders. The soldiers stationed at each castle could communicate with one another using smoke signals. This line of defense kept enemies out of Poland for years. Now the world has become smaller. Poland is growing rapidly and

the need for networking is even more dire. As Linux slowly takes root here, the goal will not be to cut off Poland from the outside and repel invaders, but to invite the community in, encouraging growth and keeping open source alive in an unlikely country.

John Biggs is a writer and consultant in Brooklyn, New York.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Optimizing Performance through Parallelism

Eric Bourque

Issue #86, June 2001

Give that tired serial algorithm an octane boost by converting it to run in SMP and distributed-memory environments.

In this article we will look at an example of how to turn a serial algorithm into one that has higher performance in symmetric multiprocessing (shared-memory), as well as distributed-memory environments. In order to fulfill this task, we will develop a simple application in three stages: a serial version, a multithreaded version and a distributed multithreaded version.

In addition to the theoretical aspects of parallel programming, we will discuss some of the practical problems encountered when programming. We have chosen to implement all of the examples in C++ and use the POSIX threads (pthreads) and MPI libraries for symmetric multiprocessing and distributed processing, respectively.

Problem Description

The problem we have chosen to explore is that of finding the number of primes in a specified range. The problem has the advantage of being both simple to comprehend and illustrative of the concepts involved with parallel programming.

Serial Implementation

In our implementation, we have chosen to represent a range of numbers by an object that has the ability to count the number of primes in its range (see Listing 1).

Listing 1. Counting Primes

Here is an example of compiling and using the program:

```
bash$ g++ -o primes primes.cpp
bash$ ./primes 0 10000
There were 1229 primes.
```

Multithreaded Implementation

In order to improve the speed of our example on symmetric-multiprocessing (SMP) machines, we need to make use of threads. We would like to stick to our design in the previous example, which means we need to find a way for each object to have its own thread of execution. Unfortunately, mixing C++ and pthreads is nontrivial, as pthread_create( ) expects a function that has been linked with C-style linking, not C++. We have solved this problem by creating an accessory class and a static member function (see Listing 2).

Listing 2. Creating an Accessory Class and a Static Member Function

The remainder of the CountPrimes object implementation is the same. There are three points to note: 1) The Threaded class is an abstract class. 2) The entry( ) function is a static member function, which means that it has knowledge of the details of the Threaded class but is not tied to a specific instance. It therefore does not go through name-mangling and can be used as a C-style function. This is the function pointer we will pass to pthread_create( ) along with a pointer to the object to be threaded. 3) The run( ) member function is *pure virtual*, and as such must be implemented by any class derived from Threaded. This function will be the main execution point of the derived class, and its return value will represent the result of the computation. In the case of our CountPrimes class, this function simply calculates the total for the range and returns it.

We would like to retain the usage form of our serial implementation, and therefore add only one extra parameter that controls the number of threads that will be spawned to complete the task. Because we do not know beforehand how many objects (threads) will be needed, we will allocate them dynamically (see Listing 3).

Listing 3. Allocation Threads

There are a few more subtleties in this example, so we will go through the code in a little more detail. First we set the default number of threads to two and then check to see if the user specified another value. We create a dynamic array of pthread_t that will store the thread ids and then create a dynamic array of pointers to CountPrimes objects. This is important because we need to instantiate each one with a different range. In fact, we could not create a static array of CountPrimes objects since there is no default constructor. This design forces us to use the object correctly.

Next is a loop that will spawn the individual threads with their respective ranges of numbers to check. Note that we have made no attempt at load balancing here. We will return to this concept later. The important point is that each CountPrimes object is instantiated dynamically, and its pointer is stored in the array created above. The thread is actually spawned through thread_create( ). We pass a pointer to the entry member function and a pointer to the object itself. The id of the spawned thread is stored in the thread id array.

Next we wait for the threads to finish computing their totals by using pthread_join( ) on the thread ids. Because we dynamically allocated space for the return value in run( ), we must de-allocate it here. Each thread's total is added to count.

Finally, the CountPrimes objects are de-alloacted, and both the thread id array and counter object pointer array are de-allocated. Here is an example of compiling and using the program:

```
bash$ g++ -o primes_threaded primes_threaded.cpp
bash$ ./primes_threaded 0 10000 4
There were 1229 primes.
```

A Distributed Implementation

Message passing interface (MPI) is a standard API for implementing distributed programs. There are many advantages of using MPI, but the main one is that programs will be compatible at the source level regardless of the particular MPI implementation being used. For the rest of this discussion, we will assume the availability of a properly configured local area multicomputer (LAM) install, an MPI implementation from Notre Dame (see Resources).

A very common model used for distributed programming is the master/slave model. In this model, there is one process called the master, which creates work and distributes it to the slaves. The slaves respond to the master with their completed work and ask for more if it is available. This conceptually simple model works very well for problems that do not require a lot of synchronization and whose slaves can be completely autonomous. These types of problems are often referred to as *embarrassingly parallel*.

In order to build on our threaded implementation, we need to decide how to reformulate our implementation in terms of a master/slave model and add the required calls to MPI in order to distribute our problem and collect the results. Listing 4 shows the changes to main( ).

Listing 4. Changes to main( )

We need to call MPI_Init( ) at the beginning of our distributed program in order to connect to the multicomputer. The next two function calls establish our rank and the total number of computers that will be involved in the computation.

MPI will start the same program on every computer in the multicomputer. This is why we need to establish at runtime what our rank is so that we can decide if we are a master or a slave. Depending on our rank, we either call master( ) or slave( ).

After we have finished our computations, we must call MPI_Finalize( ) to release our connection to the multicomputer.

Our slave( ) function takes only one argument, namely the number of threads to use. This allows us to fully utilize the processing power of SMP machines in a cluster.

The purpose of the slave is to sit and wait for work, perform the work and then return the results. It will continue to do this until it receives a signal that there is no more work to do, at which point it will return (see Listing 5).

Listing 5. Slaves

The bulk of the code in the slave( ) function is similar to main( ) in our threaded example. The only difference is how the slave gets the bounds it is supposed to count the primes in and how it returns those results.

The slave goes into an endless loop waiting for work from the master, which it gets via MPI_Recv(). This function gets two longs that are sent by the master and stores them in the bounds array. After receiving from the master, the slave checks the status of the message to see if the work is done (the KILL message), and if so, returns. Otherwise, we rename the variables so that we can use exactly the same code as in the threaded version. The only remaining step is to send our results back to the master via MPI_Send( ). Here we send back one long containing the count found by this slave.

The job of the master is slightly more complicated as it must decide how to break up the work to be sent out to the slaves and how to collect the results. The first part of the master sends the initial work units out to the slaves and waits for results to come back in. When the master receives a result, it sends another work unit out to the same process if there is still work to be done. After there is no more work to be sent out, each process is polled once more for any remaining results, and then each slave is told to quit (see Listing 6).

Listing 6. Telling Slaves to Quit

The make_work( ) function is responsible for deciding when the work is done and how to break it up. We have chosen a simple sequential model where the size of the chunks is determined by STEP_SIZE (see Listing 7).

Listing 7. Sequential Model

The STEP_SIZE variable is key to controlling the load balancing between the machines. If it is too big, there is a possibility that some machines will remain idle, while a few machines deal with the numbers in the higher end of the range. If it is too small, then there will be too much communication overhead. These factors are generally easier to determine through experimentation. These details are further explored in the Performance section.

MPI programs are compiled with **mpicc** or **mpiCC**, depending on whether you are compiling C or C++ code respectively. To run the distributed program, you must first boot the multicomputer via **lamboot**, and then you can run your program using the **mpirun** command. When you finish an MPI session, you can shut down the multicomputer with **wipe**:

```
bash$ mpiCC -O -o primes_mpi primes_mpi.cpp -lpthread
bash$ lamboot
LAM 6.3.2/MPI 2 C++/ROMIO - University of Notre Dame
bash$ mpirun -O -np 16 primes_mpi -- 0 10000000
There were 664579 primes.
bash$ wipe
```

If you are having difficulty getting lamboot to run successfully, you can use the **recon** command to verify what may be causing you trouble. If recon fails, it is possible that you are not able to run commands on remote machines without typing a password. If you are using **ssh**, make sure you have set LAMRSH to reflect that:

```
bash$ export LAMRSH=`which ssh`
```

The arguments to mpicc are essentially the same as those you would normally pass directly to your compiler. One exception is the -O to both mpicc and mpirun that specifies that the multicomputer is homogeneous and that endianness translations need not be performed. The -np argument to mpirun specifies the number of processes to start (usually the number of nodes in the multicomputer). All arguments after the double minus (**--**) are passed as arguments to the main program being run.

Performance

In order to demonstrate the effectiveness of parallel programming, we must show that the elapsed time (wall clock time) is lower for the parallel versions of our program. In general it will not be possible to get a 100% performance

increase per node, unless the problem is coarse grained and requires little synchronization.

Our tests were performed on a cluster of 16 dual PIII 700MHz with 384MB of RAM. We ran the program to calculate the number of primes between 0 and 10,000,000. Here are the times for the three versions of our program developed so far:

- Serial implementation on one node: 6:29.28 seconds.
- Multithreaded implementation on one node: 3:24.24 seconds.
- Distributed (and multithreaded) implementation on 16 nodes: 11.05 seconds.

These results show that we are getting a linear increase in performance per processor (32x speed improvement over serial version).

Load Balancing

One of the biggest problems encountered when programming a multicomputer is that of keeping each computer, and each processor in SMP computers, as busy as possible. We would like to avoid having several machines sit idle while waiting for the results of another computation being performed on a separate machine or processor. This delicate art is known as *load balancing*.

While a complete discussion of load balancing is beyond the scope of this article, we can examine a few properties of the specific problem we are solving to try to learn how to improve our performance. The single function that performs the bulk of the computation in our example is the is_prime( ) function. Due to its nature, its time is proportional to the size of the input number. Consider how we are breaking up the problem in our threaded implementation when using two threads: we send half of the numbers to one thread and the other half of the numbers to the other thread. This is inherently unbalanced because we divide the numbers sequentially. The thread with the lower half of the numbers will complete much earlier than the thread with the upper half of the numbers, and hence one processor will sit idle. There are at least two approaches to fixing this particular problem: when dividing the range of numbers, we can send every other number to each thread, or we can simply use more threads, which will break up the problem into smaller chunks and rely more on the kernel thread scheduler to balance the load. This will only work to a certain point where the time spent scheduling will exceed the gain of splitting up the problem.

There is a much more robust approach to load balancing that we used for sending jobs to machines in the distributed implementation: send out smaller

chunks of work to each machine and only send them new work when they have completed their initial work. We still need to worry a bit about the size of the chunks we send out (controlled by the STEP_SIZE variable in our implementation), or we will be increasing our network traffic without increasing our throughput. A similar approach could have been used to balance the threads but was not used for the sake of clarity.

Resources

Eric Bourque is currently a PhD candidate at McGill University in Montréal, Canada where he is researching image-based procedural texturing for computer graphics. He also holds an MSc and a BMus in Jazz Performance (saxophone) from McGill. He has taught programming courses at McGill and has done C, C++ and Perl contract teaching in various parts of Canada and the US. His company, Madison Avenue Software, specializes in custom open-source software development and programming education. Eric can be contacted at ericb@computer.org.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Web Development with PHP 4.0 and FastTemplate 1.1.0

Bill Cunningham

Issue #86, June 2001

A tutorial on the timesaving FastTemplate 1.1.0.

One thing about the web development business: the software tools we use evolve so rapidly that it's hard to keep up with them. A year ago, I hadn't even heard of the web-scripting language, PHP. Today I write PHP full-time at my job.

There we have two teams: one team writes strictly PHP code using Red Hat Linux-based PCs, and the other does strictly graphics design using DreamWeaver on NT-based PCs. The marketing department sketches out the pages; the graphics team builds the pages with the data sections of the pages stubbed out. The code team takes over and, using PHP's ability to integrate with HTML seamlessly, brings active data into the page stubs.

And that's all fine—until someone wants to change, say, the text on a certain submit button that appears on 46 pages throughout the site. This means 46 changes to 46 files now have to be made just to alter the appearance of one little button! Thank goodness for **grep**.

But there's a way to avoid all this, and that's the subject of this article. By using HTML templates, the button mentioned above can be fixed by making a simple change to one file. Templates enable the physical separation of PHP code and HTML. Changes to the PHP code in no way affect the HTML, and vice versa.

Templates operate like this: a PHP program opens and parses external files that contain the HTML for a page. The HTML files contain placeholders for data that will be substituted in at runtime. The tags usually look something like: {ITEM}. When a tag is encountered, the PHP program substitutes the right data for the tag. This way, the PHP code files can be kept completely separate from the HTML files, and the coders and designers don't accidently mess up each other's work.

There are several open-source template packages available. Some are more powerful and complicated than others. One package that I believe strikes a good balance between speed of execution and ease of use is FastTemplate 1.1.0 from CDI. Joe Harris, the author, has released FastTemplate under the General Artistic License. Joe writes in the documentation that FastTemplate 1.1.0 is actually a PHP port of a Perl module of the same name. The Perl module was written by Jason Moore.

The idea in using FastTemplate is to break down a web page into its most basic components: individual buttons, checkboxes, even lines of text. Tables must be broken down into headers and rows, and rows must be broken into cells. Each component is given its own file with the static HTML and a special tag for the variable data. Then, the page is built up from these simple components. Finally, you end up with a file like our main.tpl that simply contains the tag, {BODY}. Some prefer to call it {CONTENT}; the exact name is immaterial. This top-level tag is replaced with the data for the total page. If desired, it can even be the top-level file for every page in your site. Therefore, a change to this file will change the look of your whole site. We will do exactly that in the example program.

There are several positive implications here. To use templates, you need to have the exact content of all your pages absolutely nailed down to the last detail because you are going to be making template files for each widget on the page. This is good because it forces you to be an engineer and makes it very difficult to be a hacker. To add a feature after the fact will require changes to at least three files, not just one. Your site will be better for this because your design will be more thorough, detailed and more carefully planned.

Early on, it seems like you are accumulating a lot of very small template files, and while this is true, the templates are reusable to a large extent. As the size of your application grows, the proportion of template files shrinks. You can also put more than one data tag in a template file. This reduces the number of template files but also makes the template less reusable.

To appreciate what FastTemplate can do for web site maintainability, let's see it in action. I'm assuming that you have access to a Linux server already running Apache, PHP and a relational database (I used MySQL), and that you can make changes to a few files and directories normally owned by root.

Installing FastTemplate is about as easy as it gets. The home page is www.thewebmasters.net/php/FastTemplate.phtml. Download the file FastTemplate-1_1_0.tar.gz and, if you're running PHP 4.0, the diff file (php4.diff) as well. Move the download to your document root and untar it. The directory,

FastTemplate/, will be created. If you're running PHP 4.0, place php4.diff in FastTemplate/ and run these commands:

```
patch class.FastTemplate.php3 php4.diff
mv class.FastTemplate.php3 class.FastTemplate.php
```

Then, move class.FastTemplate.php into your PHP include directory. If you're not sure where this directory is, check the file, php.ini. It's found in /usr/local/lib/ by default. Look for the “include path” setting. It is also possible to include class.FastTemplate.php explicitly into PHP programs with an include( ) statement, but this takes extra work and I tend to forget to do it. That's it for installing FastTemplate.

Next, make another directory somewhere under your document root for this example's files. You will need the files in Listing 1, Listing 2 and Listing 3. Listing 4 will need to be broken down into the individual template files [see ftp://ssc.com/pub/lj/listings/issue86/].

Listing 1. index.php

Listing 2. mysql.php

Listing 3. goodbye.php

This example, though short and simple, illustrates that FastTemplate can easily handle a table that is dynamic in both rows and columns. As far as individual elements of a page go, that's about as complicated as it gets.

Look at Listing 1. To use FastTemplate, we simply include class.FastTemplate.php and declare an instance. The (“.”) in our declaration indicates that our supporting templates are found in the current directory. They could just as easily be in another. The FastTemplate class has four main methods: define( ), assign( ), parse( ) and print( ).

The define( ) method maps external filenames to handles that our program will use. Note that the templates need not end in .tpl. You will probably need only one call to define( ) in any program.

The assign( ) method maps the data tags (minus the {}s) to the data we want to replace them with. There can be many assign( ) calls throughout your program. Begin by dealing with the smallest, most basic components first and build more complex components with them.

For me, parse( ) was the hardest method to understand. This method parses a template, makes the data-for-tag substitution and stores the result in a local

variable. The local variable is the first parameter to parse( ). The second parameter is the file handle we defined in define( ). If the second parameter to parse( ) begins with a "." we append the new value to the previous, if any. We use this technique to populate the database selector in the example. The PHP program's main muscle will be used to retrieve or calculate data for assigning and parsing.

Finally, print( ) outputs the contents of the most recent parse by default. We can also pass parsed results to it as a parameter. There can be more than one print( ) on a page, and sometimes this is the easiest way to get a page on the screen. And, there is usually more than one way to generate a page. If it works and it's efficient, it's good.

So let's see the example. With a browser, load the examples in Listing 1, and you should see something like Figure 1. Select a database with the selector, enter a query, press the EXECUTE button and the results of your query are displayed (see Figure 2).

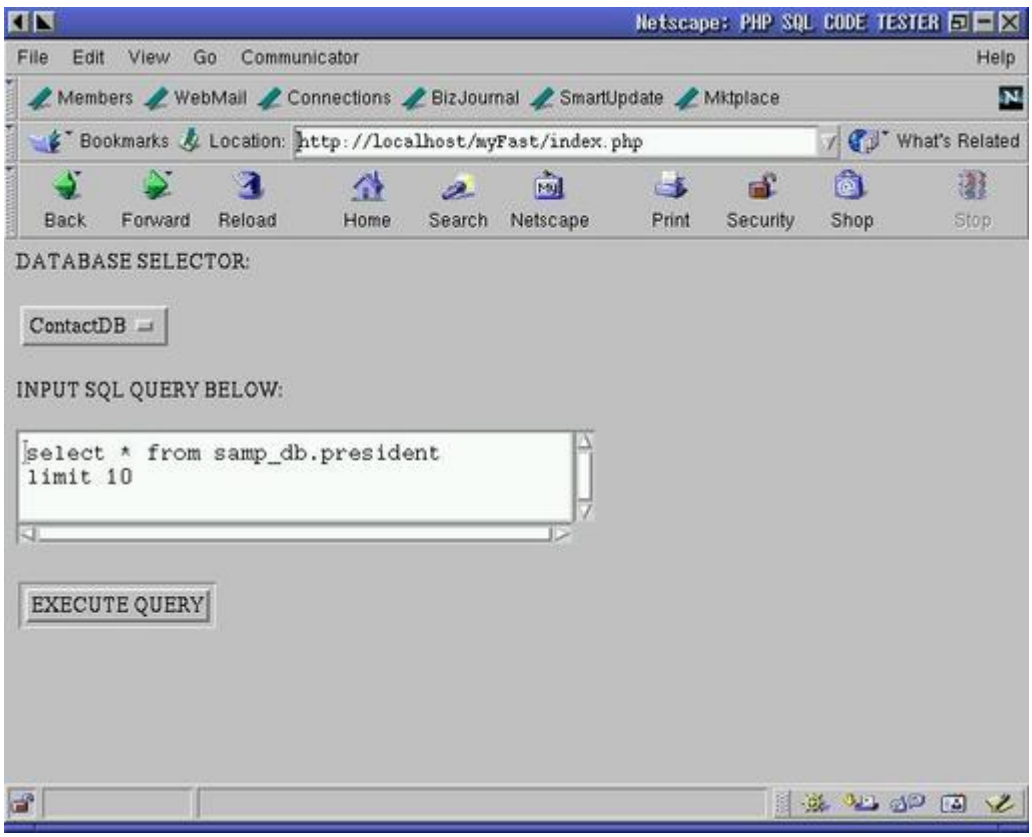


Figure 1. PHP SQL Query Test Page

last_name	first_name	suffix	city	state	birth	death
Washington	George		Wakefield	VA	1732-02-22	1799-12-14
Adams	John		Braintree	MA	1735-10-30	1826-07-04
Jefferson	Thomas		Albemarle County	VA	1743-04-13	1826-07-04
Madison	James		Port Conway	VA	1751-03-16	1836-06-28
Monroe	James		Westmoreland County	VA	1758-04-28	1831-07-04
Adams	John Quincy		Braintree	MA	1767-07-11	1848-02-23
Jackson	Andrew		Waxhaw settlement	SC	1767-03-15	1845-06-08
Van Buren	Martin		Kinderhook	NY	1782-12-05	1862-07-24
Harrison	William H.		Berkeley	VA	1773-02-09	1841-04-04
Tyler	John		Greenway	VA	1790-03-29	1862-01-18

Figure 2. Output from the Test Query

Now, assume the role of graphics designer. Your job is to totally change the look of the entire application. Your imaginary boss has just said that the grey background has to go—he wants it to be yellow. And he wants everything centered on the page, not left justified. And by the way, he wants all the text in green. With a FastTemplate-based site, you can make all this happen in less than a minute by editing the top-level template file, main.tpl.

Better web editors exist, but Netscape Composer can do the job. Here's how to do it: in Netscape Navigator, select File --> Open Page. In the dialog that appears, enter or browse the full path to main.tpl in your example directory, and click “Open in Composer”. Netscape Composer will open a large window with a single word, {BODY}, displayed in the composer edit area (see Figure 3). That's the HTML-rendered version of main.tpl. To make the boss' changes, first select Edit --> Select All. {BODY}, including parentheses, should be highlighted in yellow. Now select Format --> Align --> Center. That takes care of centering. Edit --> Select All again. Now select Format --> Text Color, and pick a nice dark green from the swatches selector. Click OK. You should now see {BODY} in dark green. That's two down, one to go. Select Format --> Page Colors and Properties. Select the Use Custom Colors button in the radiogroup at the top of the Colors and Background Tab. Then select Background. Pick yellow from the Swatches Tab in the dialog that appears. Click OK, Apply and OK again. Your background in Composer should already be yellow. Now select File --> Save, then File --> Close.

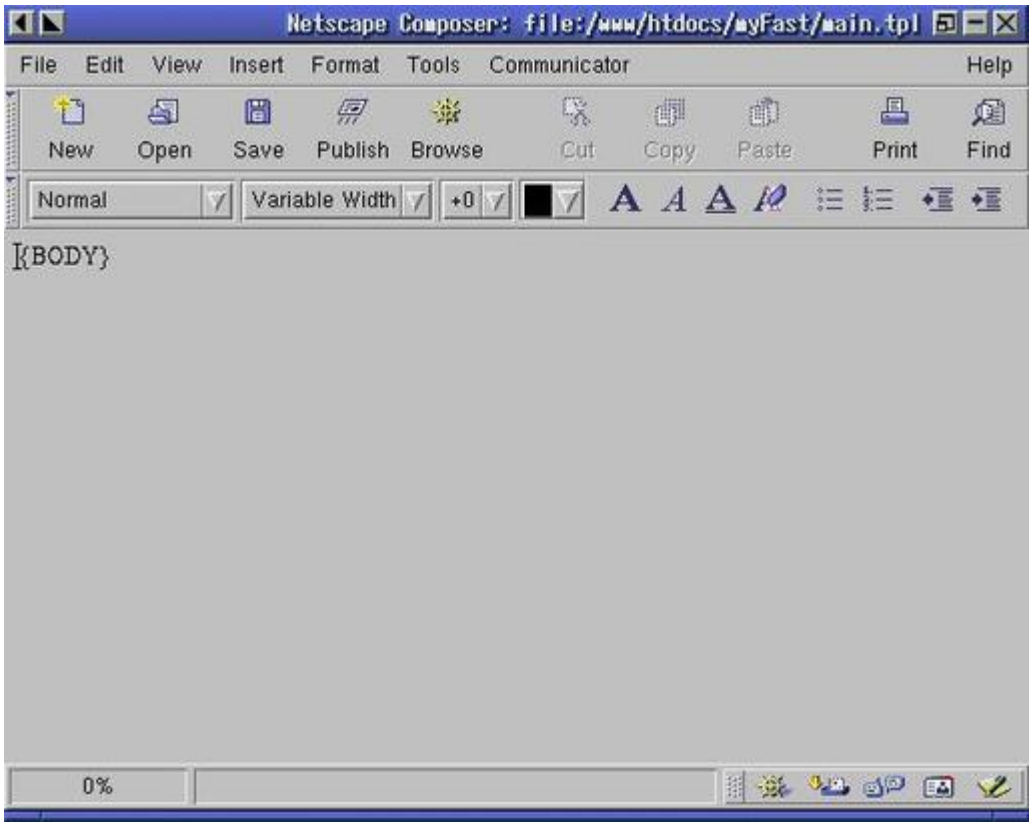


Figure 3. The Main FastTemplate File Open for Editing

Now reload the query page from the example. All three changes should now be in effect: yellow background, green text and everything centered, yet the application should still work programmatically as it always did (see Figure 4 and Figure 5). Although this example is very simple, I think the potential here is obvious.

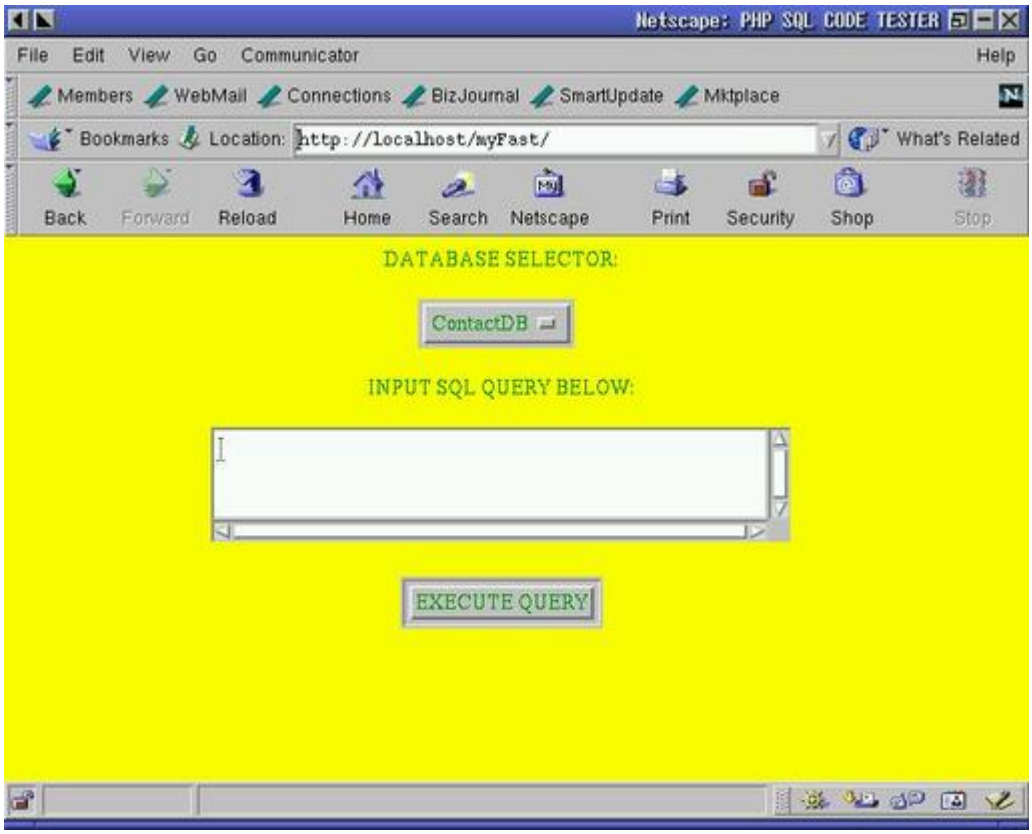
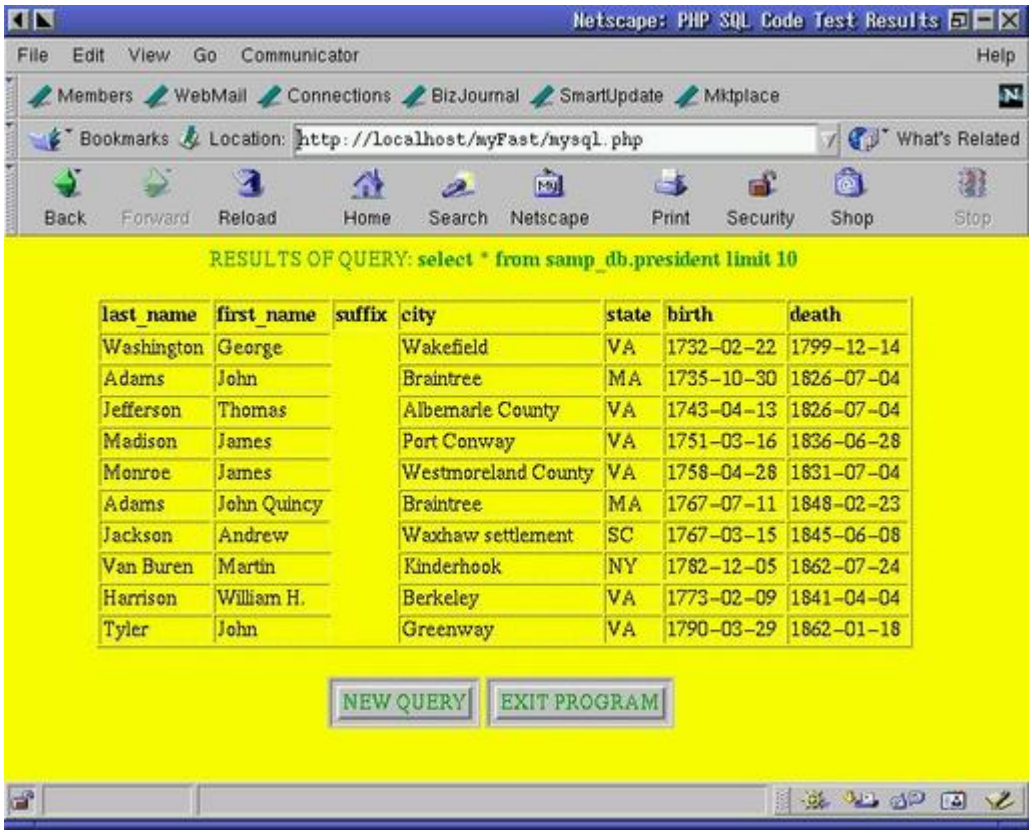


Figure 4. The New Template Is Automatically Applied to the Query Test Page



last_name	first_name	suffix	city	state	birth	death
Washington	George		Wakefield	VA	1732-02-22	1799-12-14
Adams	John		Braintree	MA	1735-10-30	1826-07-04
Jefferson	Thomas		Albemarle County	VA	1743-04-13	1826-07-04
Madison	James		Port Conway	VA	1751-03-16	1836-06-28
Monroe	James		Westmoreland County	VA	1758-04-28	1831-07-04
Adams	John Quincy		Braintree	MA	1767-07-11	1848-02-23
Jackson	Andrew		Waxhaw settlement	SC	1767-03-15	1845-06-08
Van Buren	Martin		Kinderhook	NY	1782-12-05	1862-07-24
Harrison	William H.		Berkeley	VA	1773-02-09	1841-04-04
Tyler	John		Greenway	VA	1790-03-29	1862-01-18

Figure 5. Template Applied to the Output from the Query

There are some issues that you have to consider with FastTemplate. For one, there are a lot of small files that must be managed and also read from disk

every time the page is accessed. This can cause significant performance degradation on a busy site. On sites with infrequent access, this may not be a matter for concern. In some cases, you may wish to have a single template for each page in your site to cut down on the number of template files.

Benjamin Kahn is hosting a web page for an extension of FastTemplate called Cached FastTemplate (http://zoned.net:81/~xkahn/php/fasttemplate/). Cached FastTemplate adds several performance enhancing features to FastTemplate. With this package, pages or parts of a page can be cached in memory, and the caching rules are configurable. This is worth checking out once you get comfortable with FastTemplate.

As I quickly learned at my new job, it's not enough to develop a web site that's good today. It must be scalable and maintainable, too. HTML generators like DreamWeaver, while easy to use, produce HTML that is downright ugly. You don't want to be ferreting through those files looking for your PHP code. HTML templates like FastTemplate can do a lot to enhance the quality of life for web developers.

Bill Cunningham recently retired from the US Marine Corps where he worked as a Solaris systems administrator. He is now employed by Heafner Tire Group of Charlotte, North Carolina as a web developer at a Linux/MySQL-based site.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

First Look at an Apple G4 with the Altivec Processor

Matthew Fite

Issue #86, June 2001

Matthew highlights his experience installing Linux and compiling software on his new Apple G4.

When I first read about Apple's plans to develop a G4-based personal computer, I didn't even know what a G4 was. Supercomputer performance? Processing in the GFlops? How could this be? G4, also known as the Motorola 7400, is the processor with the AltiVec unit. AltiVec is the trade name of the vector processing unit found in this new line of PowerPC processors. Motorola has also announced the 7410 and the 7450, which feature an L2 cache on the die, a large backside L3 cache, a faster processor core and a deeper, seven-stage pipeline.

The AltiVec unit is an enhanced integer or floating processing unit. It provides a new 128-bit processing unit, 32 vector registers and over 160 new instructions that allow for the processing of data in a pipeline. These provide a tremendous opportunity to move data through the processor.

After a description like that, who wouldn't want to have one of these at home? I'm not a Macintosh aficinado nor do I care for Windows very much. When I read about the work that Cort Dugan, Paul Mackerras, Ben Herrenschmidt and many others had performed porting Linux to PowerPC (PPC) I was sold. After all, this sounded like an opportunity to try something new and challenging, learn a little (or a lot) and get faster numbers from my distributed.net client (some of the reasons I started using Linux last millennium).

My hardware is an Apple dual-G4/450MHz PowerPC with 512MB RAM. It comes with a 30GB Quantum Fireball IDE drive, a CD/DVD-ROM, two IEEE-1394 (Firewire interfaces), 100Mbps Ethernet and more USB hubs than you can shake a stick at. The keyboard and mouse are both USB devices. Apple calls this a New World machine. Although this sounds like a marketing term, “New World” is used to describe Apple hardware where the boot ROM is stored in

software (as opposed to “Old World” machines where boot management software was stored in a PROM).

The Linux distribution I chose to install was Yellow Dog Linux. I don't know what finally pushed me in that direction, given that there is more than one choice—SuSE, LinuxPPC and Yellow Dog Linux immediately come to mind. YDL is based on Red Hat, so it's not too unfamiliar.

YDL is provided by Terra Soft Solutions. While Terra Soft provides another distribution, Black Lab Linux, YDL is the entry point solution for the common user. I downloaded the two ISO images of YDL Champion Server 1.2.1 from one of Terra Soft's mirrors. The first is the installation CD; the second CD is known as “Tasty Morsels”. It provides a rescue image and some additional software for the PPC. I burned these images with **cdrecord** on my SuSE/i386 box and then wondered what I had.

After I read the YDL installation guide I had some idea. The guide suggested I use **yaboot**, “yet another boot loader”. yaboot needs to live on an HFS (native Mac) partition so I needed to create one using the Mac system software.

Here are the steps I used to reinstall Mac OS9 and then install Linux:

- Create an HFS partition (4GB) for yaboot (and OS9).
- Reinstall OS9 from distribution CDs.
- From the CS 1.2.1 CD, copy yaboot, yaboot.conf and vmlinux.gz to the system folder.

yaboot.conf looks and feels a lot like lilo.conf. There are sections for each image with an area to provide a label so that when yaboot boots, the user can Tab to see the names of the kernel configurations and then select one at the prompt. Familiar stuff, but I had to modify yaboot.conf as shown below.

Here I should digress a bit about Open Firmware. Open Firmware, defined under IEEE 1275, is a specification for providing open support for firmware. This was one of the first interesting areas of my new Apple hardware explorations. I didn't see Open Firmware until I needed to. Upon booting the Mac an 880Hz tone sounds to indicates that your system just passed a hardware POST and is preparing to boot an operating system. At this point the booting process can be stopped by pressing and holding the Command-Opt-O-F keys. If all goes well the following greeting is displayed:

```
Apple PowerMac 3,3 3.4f1 BootROM built on
08/08/00 at 22:02:19
Copyright 1994-2000 Apple Computer, Inc.
Welcome to Open Firmware.
To continue booting, type "mac-boot"
and press return
```

```
To shut down, type "shut-down"
and press return
 ok
0 > _
```

The 0 > is a prompt. OF is, at its heart, a Forth interpreter. Forth is a stack-based language. To obtain a sense of this, type the following at the prompt:

```
0 > 3 [RETURN]
1 > 4 [RETURN]
2 > + [RETURN]
1 > . [RETURN]
```

and you will get the resulting:

```
0 > 7
```

The first command pushed “3” on the stack. The prompt displays the number of items on the stack before the “>”. Then I placed “4” on the stack and told the interpreter to add the results. Now there was only one item on the stack. The “.” operator pops the first value off of the stack and displays it.

You can see quite a bit about your hardware from here. For example, to see the default boot configuration of your machine type the following at the prompt:

```
0 > printenv
```

Listing 1 shows the built-in environment variables and their defaults.

Listing 1. Environment Variables and Their Defaults from the G4's Open Fireware

Another good bit of information from your PPC can be derived from the command **devalias**. Enter this command at the prompt and press the Return key. Pay attention to the value for hd. That is the hardware address of your first IDE hard drive. hd is an alias for the entire address displayed via **printenv**.

Saving This Information (More Fun with Forth)

If you are like me, you might get paranoid about changing any of these values. After a bit of research on http://developer.apple.com/, I came across some interesting technical notes. In particular were Technotes 2000-2004. Some of the benefit of having a full-featured interpreter with the power of an operating system is the ability to provide for viewing, running files and displaying hardware information for debugging. Much of this information is too detailed to write down, so there is the notion of a “two-machine” mode (TN 2004). In this mode, you can display the OF output on a serial port. The G4 PPC doesn't come with a serial port, but within Apple's OF there is a Telnet dæmon. I'm not entirely sure that you couldn't use the USB devices as output, after all, “serial” is

in the acronym, but I do know that the Telnet dæmon works. Also, I don't know if minicom can be used with a USB port.

The dæmon is easily configured. First, from the OF prompt enter the following command:

```
0 > " enet:telnet, 192.168.2.20" io
```

Observe the spaces, press Return and now OF has created a Telnet dæmon awaiting a Telnet client. This command has configured the Ethernet interface to IP address 192.168.2.20. You may want to choose a different IP address depending on your own network configuration. You will need another machine on the same physical network segment as your PPC. If you don't have a segment, a crossover Ethernet cable will do.

From your client machine, Telnet to your target (PPC) machine. You should be presented with the same “0 >” prompt as displayed from the Mac. Now you have the ability to capture all of the output from printenv, devalias, etc., to a file. This helps if you screw things up so badly that you have to return to your default configuration.

Okay, let's install Linux. Insert the YDL CD into your DVD ROM and hold the C key down while you boot. This is the method to boot from the CD. You'll be presented with the installation screen for YDL. You can follow the YDL installation guide for the most part, but a word of caution about partitioning: unless you've installed Linux before on your Mac, you'll need to create some partitions. No longer are you creating ext2 partitions, now you'll be creating partitions of the type Apple_UNIX_SVR2. Also, you'll be using **pdisk** rather than **fdisk** to create your partitions. Use the **p** command to display the partitions. If you've followed my advice above, you should see nine partitions. These are created by default, and if you intend to leave some form of running system (recommended), leave them alone.

Now you need to create the partitions for your normal partitioning scheme, I've chosen to create partitions for the mount points /, /usr, /opt, /home and a swap partition. Yours may be different, but the scheme I've created is shown in Table 1.

Table 1. Partition Map (with 512 byte Blocks) on /dev/hda

After you write the partitions to the table using the **w** command, and you quit out of pdisk (**q** command), reboot the system. pdisk will not recognize the new partitions until a reboot. Begin the installation anew by holding down the C key; indicate your newly created mount points, and you can begin selecting packages as you would on a normal Red Hat Linux installation. After you have

completed these steps, you're going to have to reboot again. This time, don't hold down any keys as you want to boot the Mac OS.

Now, back to the Mac OS. Open the yaboot.conf you copied to the system folder and take a look. Mine looks like Listing 2.

Listing 2. yaboot.conf

Notice the label for “linux”. The yaboot.conf that comes from the CD has an error; you need to prepend the extra “\\” to yaboot again. This time, use the command sequence Command-Opt-O-F to get to OF. When you again get the “0 >” prompt, enter the following:

```
0 > boot hd:,\\yaboot
```

After some flickering, you'll be presented with a LILO-like prompt. Linux should begin to boot. Success! You should now see the power of Open Firmware; the command above allows you to execute a file from your hard drive, and you haven't even booted an operating system yet!

After you log on as user root, you should edit the file /etc/modules.conf and add the following:

```
alias sound dmasound
```

This will allow you to use /dev/dsp to play audio. However, in its present form, dmasound supports write only—you can't use it to record data from an external microphone.

I configured X (XFree86 3.3.6) using the XConfigurator that runs during the Linux installation. I chose values for 1024 x 768 with a 24-bit color depth. In yaboot.conf I added the line:

```
append="video=aty128fb:vmode:17,cmode:24"
```

so that the kernel would correctly observe the ATI graphics card installed. Then I edited /etc/X11/XF86Config and added **DefaultBitsPerPixel 24** in the “Screen” section so that I didn't have to pass the bits per pixel to **startx** when I ran it.

AltiVec Stuff

Now that Linux is installed the fun with AltiVec begins. As I already mentioned, the AltiVec unit is an additional processing unit, like the floating-point unit or the integer-unit, that processes data stored in 32 128-bit vector registers. The vector execution unit processes this vector data using the single instruction

multiple data (SIMD) model. The processor, with one instruction, can operate on four, eight or 16 data units at once. Shortly I give an example to clarify this.

Motorola added 162 new assembler instructions to allow programmers to use the new functionality of the AltiVec-enabled processor. These instructions are detailed in the *AltiVec Technology Programming Environments Manual* (altivec_pem). The higher-level C instructions that use these new assembler instructions can be found in the *AltiVec Technology Programming Interface Manual* (altivec_pim). Both of these documents are available for download, in PDF format, from either Motorola's web site or from http://www.altivec.org/.

My next step was to download and install the AltiVec RPMs from http://www.altivec.org/. These RPMs provide a version of gcc (2.95.2) that has been modified to use these new directives. Installation is achieved by the following:

```
rpm -U binutils-2.9.5.0.22-6.vec.ppc.rpm
rpm -i gcc-altivec-2.95.2-1i.ppc.rpm
rpm -i gcc-altivec-c++-2.95.2-1i.ppc.rpm
```

After installation, I was able to use this new gcc as follows:

```
gcc-vec program.c -o program
```

gcc installs into /opt/bin so that it doesn't affect the default gcc. The RPM creates a link in /usr/bin, named gcc-vec, that points to the vectorized gcc in /opt.

To use the new vectorized commands, you have to write applications that use them and use a version of gcc that is aware of them. You cannot use this version of gcc on your standard C source code and expect to achieve a performance increase from the AltiVec unit. The AltiVec-enabled gcc is aware of new keywords and new functions. altivec_pim is the first step in learning the new commands provided for in gcc-vec. The new vector data types are seen in Table 2.

Table 2. AltiVec-enabled gcc: New Keywords and Functions

Notice the new keyword vector. This indicates that the following declaration is a 16-byte (128-bit) vector. Additionally, these types must be aligned on 16-byte boundaries for the vector execution unit to process the values suitably. A programmer must use caution when de-referencing data that is not aligned on a 16-byte boundary and typically will massage the data to be so aligned.

According to altivec_pim, compilers aware of the AltiVec-enabled processor should provide the following macro:

```
#define __VEC__
```

To build code that is capable of compiling on multiple architectures but is still capable of using the AltiVec instructions, you can do something like the following:

```
#ifdef __VEC__
       /* Put your vector code here */
       /* ... */
#else
       /* do it the old-fashioned way, here */
       /* ... */
#endif
```

To illustrate how to begin using the AltiVec-enabled gcc, I'll provide an example in Listing 3 [see FTP site at ftp.linuxjournal.com/pub/lj/listings/issue86].

First, notice the typedef union definitions. As previously discussed, the AltiVec registers are 128-bits. These definitions guarantee that the compiler will align the data declared by these types on 128-bit boundaries. Secondly, they provide a convenient method of accessing the individual elements of the vector data types. A final benefit of using the union data type is that now you are given a mechanism to look inside your register—by using **printf( )**. The altivec_pim provides for formatted input/output using **scanf( )/printf( )**. In theory, you should be able to print a vector float register using the following in your C code:

```
vector float f32 = (vector float)(1.1, 2.2, 3.3, 4.4);
printf( "%,vf\n", f32 );
```

To achieve this your C library (glibc*) must be aware of the vector format directives. The current implementation of the GNU C library (2.2) does not and probably never will. For this reason, I hope to modify a version of the GNU C library to serve this purpose. If you have any advice or interest, please feel free to contact me.

Next, notice the two different mechanisms for defining the vector types. The first declaration is for the vector constants stored in cVals, sVals, iVals and fVals where the vector data type is declared and defined in the same statement. This illustrates how to store constants (values that do not change at runtime) in vectors.

The next method declares a union type and assigns the vector values at runtime in an element-by-element fashion. This method would allow you to read in data from a buffer, copy it to a vector variable and pass it to your vector-aware functions.

Finally, notice the form of the vec_add( ) function. In all cases, I have used the same function, vec_add( ), and it provides the correct result, regardless of whether the arguments were vector shorts, vector ints or vector floats (the

arguments must be of the same type). In this case, the compiler interpreted the data types I passed as arguments to vec_add( ) and generated the correct form of the assembler instruction vadd* for me. For example, in the following C code the compiler is able to generate the mapping below:

```
vector float a,b,c;
/* Assign a,b */
/*    ...     */
c = vec_add( a, b);
```

This translates to the following assembler instruction:

```
vaddfp c,a,b
```

This just keeps getting easier.

To compile this program, use the following command:

```
gcc-vec -fvec vecdemo1.c -o vecdemo1
```

The -fvec switch to the compiler tells it to interpret the vector commands. If you don't use the -fvec switch, the compiler will not recognize the vector data types or commands and will print error messages that will remind you to use the switch the next time.

The program produces the output shown in Listing 4.

Listing 4. Output of the Vecdemo1 Command

I've tried to provide an introduction to Linux on the PowerMac and to the AltiVec resources available to Linux programmers. I would like to do more. Other possible avenues would be to demonstrate how the AltiVec can be used as a platform for signal processing, how these processors can be used in place of special-purpose DSPs or to look at a common use for DSPs in signal processing, finite impulse response (FIR) filters.

I would like to thank the members of the AltiVec forum. The mail list has been an invaluable resource to get up and running. Also, thank you to all of the AltiVec developers that have provided such a rich set of tools to begin development on a platform as powerful as the G4.

Resources

Matthew Fite and his wife live in northern Virginia where he works as an embedded software engineer. Although he uses a commercial RTOS during the day, he secretly dreams of replacing it with RTLinux. Matthew has a BS and MS in Electrical Engineering and is always looking for a new project, although his wife probably believes he has enough. You can contact him at mattfite@yahoo.com.

Archive Index Issue Table of Contents

Advanced search

The Linux Socket Filter: Sniffing Bytes over the Network

Gianluca Insolvibile

Issue #86, June 2001

A feature added to the kernel with the 2.2 release, this LSF can be programmed to let the kernel decide to which packets access should be granted. Here's how.

If you deal with network administration or security management, or if you are merely curious about what is passing by over your local network, grabbing some packets off the network card can be a useful exercise. With a little bit of C coding and a basic knowledge of networking, you will be able to capture data even if it is not addressed to your machine. In this article, we will refer to Ethernet networks, by far the most widespread LAN technology. Also, for reasons that will be explained later, we will assume that source and destination hosts belong to the same LAN.

First off, we will briefly recall how a common Ethernet network card works. Those of you who are already skilled in this field may safely skip to the next paragraph. IP packets sourced from users' applications are encapsulated into Ethernet frames (this is the name given to packets when sent over an Ethernet segment), which are just bigger lower-level packets containing the original IP packet and some information needed to carry it to its destination (see Figure 1). In particular, the destination IP address is mapped to a 6-byte destination Ethernet address (often called MAC address) through a mechanism called ARP. Thus, the frame containing the packet travels from the source host to the destination host over the cable that connects them. It is likely that the frame will go through network devices such as hubs and switches, but since we assumed no LAN borders are crossed, no routers or gateways will be involved.

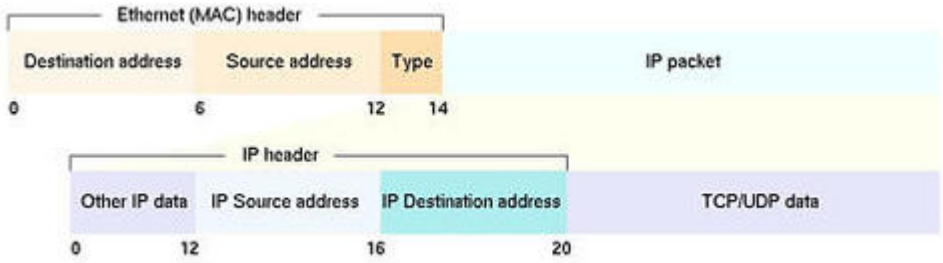


Figure 1. IP Packets as Ethernet Frames

No routing process happens at the Ethernet level. In other words, the frame sent by the source host will not be headed directly toward the destination host; instead, the frame will be copied over all the cables that make up the LAN, and all the network cards will see it passing (see Figure 2). Each network card will start reading the first six bytes of the frame (which happen to contain the above-mentioned destination MAC addresses), but only one card will recognize its own address in the destination field and will pick up the frame. At this point, the frame will be taken apart by the network driver and the original IP packet will be recovered and passed up to the receiving application through the network protocol stack.

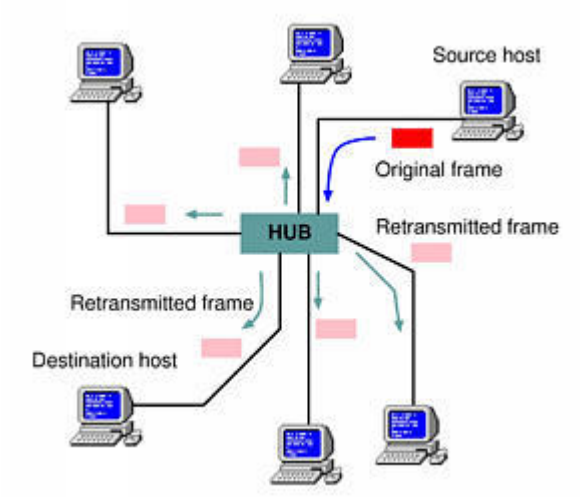


Figure 2. Sending Ethernet Frames over the LAN

More precisely, the network driver will have a look at the Protocol Type field inside the Ethernet frame header (see Figure 1) and, based on that value, forward the packet to the appropriate protocol receiving function. Most of the time the protocol will be IP, and the receiving function will take off the IP header and pass the payload up to the UDP- or TCP-receiving functions. These protocols, in turn, will pass it to the socket-handling functions, which will eventually deliver packet data to the receiving application in userland. During this trip, the packet loses all network information related to it, such as the source addresses (IP and MAC) and port, IP options, TCP parameters and so on. Furthermore, if the destination host does not have an open socket with the

correct parameters, the packet will be discarded and never make it to the application level.

As a consequence, we have two distinct issues in sniffing packets over the network. One is related to Ethernet addressing—we cannot read packets that are not destined to our host; the other is related to protocol stack processing—in order for the packet not to be discarded, we should have a listening socket for each and every port. Furthermore, part of the packet information is lost during protocol stack processing.

The first issue is not fundamental, since we may not be interested in other hosts' packets and may tend to sniff all the packets directed to our machine. The second one, however, must be solved. We will see how to address these issues separately, starting with the latter.

The PF_PACKET Protocol

When you open a socket with the standard call **sock = socket(domain, type, protocol)** you have to specify which domain (or protocol family) you are going to use with that socket. Commonly used families are PF_UNIX, for communications bounded on the local machine, and PF_INET, for communications based on IPv4 protocols. Furthermore, you have to specify a type for your socket and possible values depend on the family you specified. Common values for type, when dealing with the PF_INET family, include SOCK_STREAM (typically associated with TCP) and SOCK_DGRAM (associated with UDP). Socket types influence how packets are handled by the kernel before being passed up to the application. Finally, you specify the protocol that will handle the packets flowing through the socket (more details on this can be found on the socket(3) man page).

In recent versions of the Linux kernel (post-2.0 releases) a new protocol family has been introduced, named PF_PACKET. This family allows an application to send and receive packets dealing directly with the network card driver, thus avoiding the usual protocol stack-handling (e.g., IP/TCP or IP/UDP processing). That is, any packet sent through the socket will be directly passed to the Ethernet interface, and any packet received through the interface will be directly passed to the application.

The PF_PACKET family supports two slightly different socket types, SOCK_DGRAM and SOCK_RAW. The former leaves to the kernel the burden of adding and removing Ethernet level headers. The latter gives the application complete control over the Ethernet header. The protocol field in the socket() call must match one of the Ethernet IDs defined in /usr/include/linux/if_ether.h, which represents the registered protocols that can be shipped in an Ethernet frame. Unless dealing with very specific protocols, you typically use ETH_P_IP,

which encompasses all of the IP-suite protocols (e.g., TCP, UDP, ICMP, raw IP and so on).

Since they have pretty serious security implications (for example, you may forge a frame with a spoofed MAC address), PF_PACKET-family sockets may only be used by root.

The PF_PACKET family easily solves the problem associated with protocol stack-handling of our sniffed packets. Let's see it do so with the example in Listing 1. We open a socket belonging to the PF_PACKET family, specifying a SOCK_RAW socket type and IP-related protocol type. Then we start reading from the socket and, after a few sanity checks, we print out some information extracted from the Ethernet level and IP level headers. By cross-checking the printed addresses with the offsets in Figure 1, you will see how easy it is for the application to get access to network level data.

Listing 1. Protocol Stack-Handling Sniffed Packets

Assuming that your machine is connected to an Ethernet LAN, you can experiment with our short example by running it while generating packets directed to your host from another machine (you can **ping** or Telnet to your host). You will be able to see all the packets directed to you, but you will not see any packet headed toward other hosts.

Promiscuous vs. Nonpromiscuous Mode

The PF_PACKET family allows an application to retrieve data packets as they are received at the network card level, but still does not allow it to read packets that are not addressed to its host. As we have seen before, this is due to the network card discarding all the packets that do not contain its own MAC address—an operation mode called nonpromiscuous, which basically means that each network card is minding its own business and reading only the frames directed to it. There are three exceptions to this rule: a frame whose destination MAC address is the special broadcast address (FF:FF:FF:FF:FF:FF) will be picked up by any card; a frame whose destination MAC address is a multicast address will be picked up by cards that have multicast reception enabled and a card that has been set in promiscuous mode will pick up all the packets it sees.

The last case is, of course, the most interesting one for our purposes. To set a network card to promiscuous mode, all we have to do is issue a particular ioctl() call to an open socket on that card. Since this is a potentially security-threatening operation, the call is only allowed for the root user. Supposing that “sock” contains an already open socket, the following instructions will do the trick:

```
strncpy(ethreq.ifr_name,"eth0",IFNAMSIZ);
ioctl(sock, SIOCGIFFLAGS, &ethreq);
ethreq.ifr_flags |= IFF_PROMISC;
ioctl(sock, SIOCSIFFLAGS, &ethreq);
```

(where ethreq is an ifreq structure, defined in /usr/include/net/if.h). The first ioctl reads the current value of the Ethernet card flags; the flags are then ORed with IFF_PROMISC, which enables promiscuous mode and are written back to the card with the second ioctl.

Let's see it in a more complete example (see Listing 2 at ftp://ftp.linuxjournal.com/pub/lj/listings/issue86/). If you compile and run it as root on a machine connected to a LAN, you will be able to see all the packets flowing on the cable, even if they are not for your host. This is because your network card is now working in promiscuous mode. You can easily check it out by giving the **ifconfig** command and observing the third line in the output.

Note that if your LAN uses Ethernet switches instead of hubs, you will see only packets flowing in the switch's branch you belong to. This is due to the way switches work, and there is very little you can do about it (except for deceiving the switch with MAC address-spoofing, which is outside the scope of this article). For more information on hubs and switches, please have a look at the articles cited in the Resources section.

The Linux Packet Filter

All our sniffing problems seem to be solved right now, but there is still one important thing to consider: if you actually tried the example in Listing 2, and if your LAN serves even a modest amount of traffic (a couple of Windows hosts will be enough to waste some bandwidth with a good number of NETBIOS packets), you will have noticed our sniffer prints out too much data. As network traffic increases, the sniffer will start losing packets since the PC will not be able to process them quickly enough.

The solution to this problem is to filter the packets you receive, and print out information only on those you are interested in. One idea would be to insert an “if statement” in the sniffer's source; this would help polish the output of the sniffer, but it would not be very efficient in terms of performance. The kernel would still pull up all the packets flowing on the network, thus wasting processing time, and the sniffer would still examine each packet header to decide whether to print out the related data or not.

The optimal solution to this problem is to put the filter as early as possible in the packet-processing chain (it starts at the network driver level and ends at the application level, see Figure 3). The Linux kernel allows us to put a filter, called an LPF, directly inside the PF_PACKET protocol-processing routines, which are

run shortly after the network card reception interrupt has been served. The filter decides which packets shall be relayed to the application and which ones should be discarded.

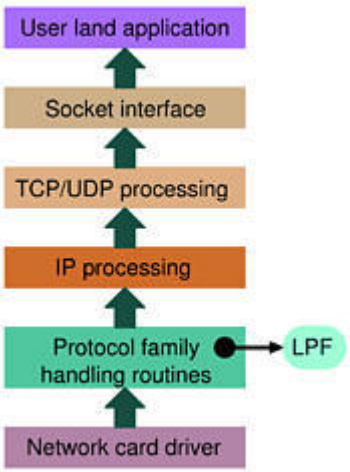


Figure 3. Packet-Processing Chain

In order to be as flexible as possible, and not to limit the programmer to a set of predefined conditions, the packet-filtering engine is actually implemented as a state machine running a user-defined program. The program is written in a specific pseudo-machine code language called BPF (for Berkeley packet filter), inspired by an old paper written by Steve McCanne and Van Jacobson (see Resources). BPF actually looks like a real assembly language with a couple of registers and a few instructions to load and store values, perform arithmetic operations and conditionally branch.

The filter code is run on each packet to be examined, and the memory space into which the BPF processor operates are the bytes containing the packet data. The result of the filter is an integer number that specifies how many bytes of the packet (if any) the socket should pass to the application level. This is a further advantage, since often you are interested in just the first few bytes of a packet, and you can spare processing time by avoiding copying the excess ones.

(Not) Programming the Filter

Even if the BPF language is pretty simple and easy to learn, most of us would probably be more comfortable with filters written in human-readable expressions. So, instead of presenting the details and instructions of the BPF language (which you can find in the above-mentioned paper), we will discuss how to obtain the code for a working filter starting from a logic expression.

First, you will need to install the **tcpdump** program from LBL (see Resources). But, if you are reading this article, it is likely that you already know and use

tcpdump. The first versions were written by the same people who wrote the BPF paper and its first implementation. In fact, tcpdump uses BPF, in the form of a library called libpcap, to capture and filter packets. The library is an OS-independent wrapper for the BPF engine. When used on Linux machines, BPF functions are carried out by the Linux packet filter.

One of the most useful functions provided by the libpcap is pcap_compile(), which takes a string containing a logic expression as input and outputs the BPF filter code. tcpdump uses this function to translate the command-line expression passed by the user into a working BPF filter. What is interesting for our purposes is that tcpdump has a seldomly used switch, -d, which prints the code of the filter.

For example, typing **tcpdump host 192.168.9.10** will start sniffing and grab only those packets whose source or destination IP address matches 192.168.9.10. Typing **tcpdump -d host 192.168.9.10** will print the BPF code that recognizes the filter, as shown in Listing 3.

Listing 3. Tcpdump -d Results

Let's briefly comment on this code; lines 0-1 and 6-7 verify that the captured frame is actually transporting IP, ARP or RARP protocols by comparing their protocol IDs (see /usr/include/linux/if_ether.h) with the value found at offset 12 in the frame (see Figure 1). If the test fails, the packet is discarded (line 13).

Lines 2-5 and 8-11 compare the source and destination IP addresses with 192.168.9.10. Note that, depending on the protocol, the offsets of these addresses are different; if the protocol is IP, they are 26 and 30, otherwise they are 28 and 38. If one of the addresses matches, the packet is accepted by the filter, and the first 68 bytes are passed to the application (line 12).

The filter code is not always optimized, since it is generated for a generic BPF machine and not tailored to the specific architecture that runs the filter engine. In the particular case of the LPF, the filter is run by the PF_PACKET processing routines, which may have already checked the Ethernet protocol. This depends on the protocol field you specified in the initial socket() call: if it is not ETH_P_ALL (which means that every Ethernet frame shall be captured), then only frames having the specified Ethernet protocol will arrive at the filter. For example, in the case of an ETH_P_IP socket, we could rewrite a faster and more compact filter as follows:

```
(000) ld       [26]
(001) jeq      #0xc0a8090a      jt 4    jf 2
(002) ld       [30]
(003) jeq      #0xc0a8090a      jt 4    jf 5
(004) ret      #68
(005) ret      #0
```

Installing the Filter

Installing an LPF is a straightforward operation: all you have to do is create a sock_filter structure containing the filter and attach it to an open socket.

The filter structure is easily obtained by substituting the -d switch to tcpdump with -dd. The filter will be printed as a C array that you can copy and paste into your code, as shown in Listing 4. Afterward, you attach the filter to the socket by simply issuing a setsockopt() call.

Listing 4. Tcpdump with --dd Switch

A Complete Example

We will conclude this article with a complete example (see Listing 5 at ftp://ftp.linuxjournal.com/pub/lj/listings/issue86/). It is exactly like the first two examples, with the addition of the LSF code and the setsockopt() call. The filter has been configured to select only UDP packets, having either source or destination IP address 192.168.9.10 and source UDP port equal to 5000.

In order to test this listing, you will need a simple way to generate arbitrary UDP packets (such as sendip or apsend, found on http://freshmeat.net/). Also, you may want to adapt the IP address to match the ones used in your own LAN. To accomplish this, just substitute 0xc0a8090a in the filter code with the IP address of your choice in hex notation.

A final remark concerns the status of the Ethernet card when you exit the program. Since we did not reset the Ethernet flags, the card will remain in promiscuous mode. To solve this problem, all you need to do is install a Control-C (SIGINT) signal handler that resets the Ethernet flags to their previous value (which you will have saved just before ORing with IFF_PROMISC) before exiting the program.

Conclusions

Sniffing packets over your LAN is an invaluable tool for debugging network problems or collecting measurements. Sometimes the commonly available tools, such as tcpdump or **ethereal**, will not exactly fit your needs and writing your own sniffer can be of great help. Thanks to the LPF, you can do this in a simple and efficient way.

Resources

Gianluca Insolvibile has been a Linux enthusiast since kernel 0.99pl4. He currently deals with networking and digital video research and development. He can be reached at g.insolvibile@cpr.it.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

JavaBeans

Reuven M. Lerner

Issue #86, June 2001

Using beans makes working with JSPs easier for all levels of programmers.

In the previous two installments of “At the Forge”, we began to work with Jakarta-Tomcat, an open-source servlet and Java Server Pages (JSP) engine sponsored by the Apache Software Foundation. As we have seen, it is neither difficult nor time-consuming to create servlets or JSPs. Getting used to the server-side Java paradigm is probably the biggest hurdle to using them.

While servlets offer us the full breadth of power and expressiveness available from within a Java program, they force us to think at a relatively low level. Every time we want to send HTML-formatted text to the user's browser—quite often, normally—we must use the PrintWriter object associated with the HTTP response object:

```
PrintWriter out = response.getWriter();
out.println("<HTML><Body>This is illegal HTML</Body></HTML>");
```

JSPs come to the rescue, assuming that everything that is not explicitly marked as an executable code section should be sent verbatim to the user's browser. But this creates a new problem, namely the fact that a JSP wanting to connect to a relational database must add dozens or hundreds of lines of Java code.

The solution is to create bundles of code that reside outside of the JSP, whose methods can be invoked using a syntax that resembles HTML more than it does Java. These bundles of code are known as JavaBeans, and they can make it much easier to work with JSPs, both for experienced programmers who want to work at a higher level, and for inexperienced programmers who want to take advantage of functionality.

This month, we will take a quick tour of JavaBeans. We will write some of our own beans, integrate them into JSPs and discuss some of the problems and pitfalls associated with them.

What Is a Bean?

From the perspective of someone implementing a bean, JavaBeans are nothing more than Java classes that adhere to several conventions. (We will soon discuss just what those conventions are.)

But to someone who writes a JSP, a bean is a special type of container into which we can store and retrieve certain types of information. Each piece of information is known as a property and can be set or retrieved individually. Not all properties can be set and not all of them can be retrieved, but the interface to the bean from a JSP is uniform and easy to understand.

Because beans understand a restricted set of actions, there are special JSP tags that allow us to work with them. Using these tags allows us to reduce the amount of Java code directly placed within our JSPs. Not only does this reduce clutter and make our JSPs more maintainable and readable, but it means that nonprogrammers can take advantage of a bean's power without having to learn to program in Java.

It's not unusual for a JSP to use multiple beans simultaneously, storing and retrieving different properties as necessary. Thus, a JSP for an on-line store's shopping cart might use one bean for the store's inventory, another for the user's shopping cart, and still another to track the user's language, payment and shipping preferences. Each of these beans is implemented by a separate Java class but is manipulated using special JSP tags that hide most of the complexity from the JSP author.

Of course, a bean can be used on multiple web sites (or by multiple portions of a single site). If you develop a useful JavaBean that encapsulates interesting functionality, other users can drop that bean into their Java classpath, taking advantage of the functionality from within their JSPs.

Implementing a Bean

To write a bean, we must write a Java class that implements the java.io.Serializable interface. Simply put, this means that a bean must be able to save to disk and restore itself. If the fields of your class are common Java types, such as integers and strings, then implementing Serializable shouldn't concern you too much.

Listing 1 contains a simple bean implementation. This bean contains a single instance variable (userID) and two methods. The getUserID method returns the current value of userID, while the setUserID method sets the value of that field. Because these methods' return values and parameter lists match the signatures for bean property methods, we can use them from within our JSPs.

Listing 1. SimpleBean.java

On my system, which is running version 3.2 of the Apache project's Jakarta-Tomcat servlets/JSP system, I placed my Java classes under the directory $TOMCAT_HOME/classes ($TOMCAT_HOME is an environment variable that points to the root of the Tomcat installation; on my system, its value is /usr/java/jakarta-tomcat-3.2.1/). If this “classes” directory exists, it is added to the Tomcat CLASSPATH environment variable, making it a convenient place to put new classes.

The class itself is very simple, demonstrating the different types of methods you can create: 1) A bean constructor that takes no arguments and can set one or more fields. In our particular example, the SimpleBean constructor initializes userID to be 0. 2) A get property method that returns a value to the caller. Like the bean constructor, a get property method does not take any arguments. 3) A set property method that takes a single argument (the new value) but does not return any value to its caller.

Keep in mind that beans are classes like most other Java classes, meaning that they must be recompiled before they are reused. Moreover, the Tomcat servlet container does not automatically reload classes that have been compiled. Your best bet is to restart Tomcat each time you recompile a bean class.

Using the Bean

Now that we have created our simple bean class, how do we use it from within a JSP? JSPs recognize three special tags, all of which begin with “jsp:”.

Before we continue, a word of warning: the special tags that allow us to work with JavaBeans from within a JSP are written in XML rather than HTML. While HTML is a loosely defined specification that does not always require us to close a tag, XML is much stricter. Every opening <tag> must be closed by a matching </tag>. Without a closing </tag>, the XML parser will exit with an error. A tag can close itself by placing a slash at the end of the tag, as in <tag/>.

This means that within an HTML-generating JSP using JavaBeans, you will have to keep track of two slightly different syntaxes. After a while, you will find that moving between these two syntaxes is almost second nature. Moreover, the JSP parser produces error messages that make it relatively easy to determine when you have forgotten to place a trailing slash on a JavaBean tag. However, this mix of syntaxes can be maddening at the beginning, and you can expect to go through some hardship in learning to understand it.

To use a JavaBean class, we use the special <jsp:useBean/> tag. This tag tells the JSP to find and load a particular bean and to create an instance of the bean

within our JSP. The <jsp:useBean/> tag also lets us give our bean instance a name that we will use later on. Here is an example of how to load our SimpleBean class from within a JSP:

```
<jsp:useBean id="simple"
 class="il.co.lerner.SimpleBean"/>
```

As you can see, the tag ends with a slash (/), following the XML syntax. There are actually cases in which you might prefer to separate the <jsp:useBean/> tag into an opening <jsp:useBean> and closing </jsp:useBean>; whatever is between those two tags is only performed when the bean is first loaded into memory. However, the simpler form is not uncommon.

The <jsp:useBean/> tag takes two mandatory parameters. The class parameter names the package and class in which our bean sits. The id parameter gives our bean a unique name within the JSP. As with variable names, it is a good idea to choose clear identifiers for working with beans. The more obvious the name, the easier it will be to debug our JSP later on.

We can set and retrieve property values within our bean instance with the <jsp:setProperty/> and <jsp:getProperty/> tags. Both of these tags take a name parameter whose value should be identical to the id that we gave our bean earlier on (I am sure there is a good reason why we use an id attribute in <jsp:useBean/> and a name attribute in <jsp:setProperty/>, but I find it to be confusing). We can thus retrieve the value of the userID property with the following tag:

```
<jsp:getProperty name="simple" property="userID"/>
```

This retrieves the userID property from the simple bean and places it inside of the JSP. Note that this does not simply retrieve the value, it also makes it visible to the user. Also notice that the capitalization of our property name has been altered slightly. In order to access the getUserID method within our bean, we use <jsp:getProperty/> for the property userID. Don't be fooled into thinking the property names are case-insensitive, however; this transformation takes place simply to keep things readable.

To modify the value of a property, we use the <jsp:setProperty/> tag. This tag does not return any results, but it does take a value attribute whose value is then handed to the appropriate method in the bean class:

```
<jsp:setProperty name="simple" property="userID"
 value="300"/>
```

Listing 2 contains a complete JSP that demonstrates how we can use our SimpleBean class from within a JSP. It displays the default value of the userID property, then sets that property to a new value and displays the new value.

Listing 2. use-simple.jsp

Parameters and Properties

It is certainly common for properties to be stored in a bean's instance variables, as in our SimpleBean class. In such cases, invoking <jsp:setProperty/> effectively sets the value of the field, and invoking <jsp:getProperty/> retrieves its current value. Of course, there is no reason why properties must reflect fields. Properties can easily be stored to and retrieved from a relational database. When you retrieve a property, it is possible the returned value is being calculated in real time rather than being returned from an instance variable.

Consider, for example, how we might create a bean that performs simple mathematical operations. We can set two read/write properties (call them arg1 and arg2) and then a number of read-only properties we can use to perform calculations on these arguments. Listing 3 (available at ftp.linuxjournal.com/pub/lj/listings/issue86) contains a simple bean, Calculate.java, which demonstrates how we can accomplish this.

Because there is no setSum property, the JSP engine will not allow us to invoke <jsp:setProperty/> on the sum property. However, it will allow us to set arg1 and arg2 and to retrieve each of the individual properties we might want.

Now that we have a working bean, we can use it from within a JSP. Listing 4 contains the listing for calculator.jsp, which performs some basic calculations using the JavaBean we just created.

Listing 4. calculator.jsp

The first and most interesting part of calculator.jsp is the way in which it sets the properties:

```
<jsp:setProperty name="calculator" property="*"/>
```

Normally, we can take one of the parameters passed via GET or POST and assign its value to a particular property using the following notation:

```
<jsp:setProperty name="calculator"
 parameter="foo" property="arg1"/>
```

In other words, the above tag will take the foo parameter and use its value when invoking setArg1 on the calculator bean. But when we use an asterisk, as in Listing 4, we indicate to the JSP engine that we want to take each of the parameters we received and assign each of the values to the properties of the

same name. Thus, the arg1 parameter will be passed to the arg1 property and so forth.

On my system, where I have installed calculator.jsp under the examples/jsp URL, I can assign arg1 the value 5 and arg2 the value 20 with the following URL: http://localhost/examples/jsp/calculator.jsp?arg1=5&arg2=20.

Of course, this is not a foolproof system. I can cause a runtime exception by passing the following URL: http://localhost/examples/jsp/calculator.jsp?arg1=5&arg2=20.0.

The JSP engine tries to assign 20.0 (a float value) to arg2 (an int), which fails.

We can surround our <jsp:setProperty/> and <jsp:getProperty/> tags with scriptlet tags, using the standard Java try-and-catch mechanism to attempt to avoid runtime errors. For example, the following code ensures we will never have to deal with division-by-zero exceptions when using getQuotient:

```
<P>Quotient:
    <% try { %>
        <jsp:getProperty name="calculator"
        property="quotient"/>
    <% } catch (Exception e) { %>
        <B>Error! division by zero</B>
    <% } %>
</P>
```

At the same time, this sort of code introduces more Java into the JSP, precisely the reason why we began to use beans. Whether the nonprogrammers on your staff will be able to handle this sort of code depends very much on your environment, as well as the sorts of beans you have created.

Bean Scopes

As we have discussed in previous months, the servlet container loads only a single copy of every servlet into memory at a given time. It can get away with this because Java is multithreaded, allowing a particular servlet to be executing simultaneously for several HTTP requests. Since JSPs are actually servlets in disguise, they are also subject to issues of multithreading.

This raises the question of what happens to our JavaBeans: how many times are they loaded; what is their scope? If a JSP sets a property that happens to modify a class' fields, does this affect all of the other instances of the same bean?

The answer is: It depends. There are four different types of bean scopes, and the type of scope that you choose will profoundly affect the way your bean is used. Application scope means a single copy of the bean for all JSPs is running

within the servlet container. Session scope is for a single user, from the time they enter the site to the time they exit. If a user opens two browser windows onto your system, they will have identical sessions. Request scope extends through the end of an HTTP request. This is useful if you want to set a bean's properties in one JSP, use the <jsp:forward/> tag to perform an internal redirect to a second JSP and then continue to use that same bean from the second JSP. Page scope is for a single JSP page. When the page exits, so does the scope.

By default, beans are placed in the session scope. You can change this by adding a scope parameter to the <jsp:useBean/>:

```
<jsp:useBean id="simple" scope="application"
 class="il.co.lerner.SimpleBean"/>
```

Once we have done the above, properties set by one JSP will be visible to other JSPs. Of course, because beans in the application and server scopes might be executed by more than one JSP simultaneously, they must be threadsafe. Consider the scopes in which your beans are meant to be used, and make them threadsafe as necessary. If a bean is not threadsafe, be sure to indicate as much in its documentation so that others will not mistakenly use it in the wrong way.

Web Log

Last month, we continued our simple investigations of web logs with a JSP that directly accessed a relational database to get the latest contents of a web log. While such a JSP is certainly legal, it looks awkward, is difficult to debug and fails to separate code from logic as elegantly as we might have hoped.

By putting the database logic into a JavaBean, we can achieve several of our goals: the code is reusable, available even to nonprogrammers and allows us to change the source of information or the application logic without rewriting our JSP. Listing 5 (available at ftp.linuxjournal.com/pub/lj/listings/issue86) contains the source code for our bean, which I have made threadsafe (using the “synchronized” keyword within the getBlog method) so we can create a single instance of application scope. This JavaBean connects to the web log database and retrieves the latest information. Listing 6 contains a small JSP that uses this JavaBean to display the web log.

Listing 6. viewblog.jsp

There really isn't anything special about the bean in Listing 5, but it does bring together a number of things that we have been discussing in the last few months. We now have a way for nonprogrammers to access information in our web log without having to write a single line of code on their own! Using a few

simple JSP tags, we can place our current blog contents in an HTML page quickly and easily.

Conclusion

JavaBeans are a wonderful way in which to reduce the amount of code we put in our JSPs, while making it easier to use. However, complex JSPs will still contain some code, such as when they have to iterate through loops or work with complex data.

Next month, we will see how we can write our own tags similar to the jsp: tags we saw this month, using JSP's custom actions facility. Therefore, we can create our own new tags, associating them with any code we wish. In this way, JSPs allow us to create our own new formatting language, as well as use the tags that have been provided.

Resources

Reuven M. Lerner owns and manages a small consulting firm specializing in web and internet technologies. As you read this, he should be (finally) finishing Core Perl, to be published by Prentice-Hall later this year. You can reach him at reuven@lerner.co.il, or at the ATF home page, http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

A Taste of the World

Marcel Gagné

Issue #86, June 2001

Marcel and his faithful assistant offer their usual unique spin on this month's feature topic.

François, *mon ami*. I take it from your smiling face you have completed your menu suggestions. I must tell you, *mon ami*, I was a little worried about letting you do this. It is not easy for this chef to let his waiter come up with the recipes for an entire issue, *non*? Come here, and let me see what you have chosen.

Mon Dieu! As I have told you many times, *mon ami*, you are a wonderful waiter, but sometimes...to tell the truth, *mon ami*, you do not know what is meant by "internationalization and emerging markets", do you? Aha, I did not think so. Why did you not ask? Well, for one thing, emerging markets does not refer to stocks. François, why are you not paying attention? What are you looking at?

Ah, *mes amis*. Welcome to *Chez Marcel*. Please, sit down. Be comfortable. I will have François fetch some wine, *immédiatement*. François, since we are talking about the world, why not offer *nos amis* a taste of Tuscany. Please, bring up the 1990 Olmaia from the cellar and do not fret, all will be well.

Non, *mes amis*, do not worry. It's just that François wanted to come up with the menu for this issue on internationalization and emerging markets, and I fear he should have asked what those terms meant. He does have some interesting items on the menu, even if his understanding is a bit off. To start with, he has come up with a wonderfully simple little stock ticker program written by Tom Poindexter. It is a Tcl/Tk program and requires version 8.0 of Tcl/Tk or later. Other than that, it is a breeze to set up. Start by getting the latest source from www.nyx.net/~tpoindex/tcl.html#Tclticker, and unpack it in a temporary directory. There is one other small thing I should mention. You must have an internet connection since that is where the information comes from, *non*?

```
tar -xzvf tclticker-1.2.tar.gz
cd tclticker-1.2
```

Since this is a Tcl script, you do not need to compile anything. In fact, you can use this as an example to explore Tcl programming. By default, the program keeps its information in the /lib directory directly beneath the home of the executable. Knowing that, I run the program right from where it was extracted.

```
./tckticker &
```

Figure 1. Keeping Track of Your Stocks with TckTicker

Wonderful, *non*? And so simple. To add (or remove) stocks or indexes from the list you are watching, simply click on the ticker and a Set Parameters window will appear with a list of stock symbols that you can easily modify. There is even a simple symbol lookup for when you know the name of the company but nothing else. You can also change the speed at which the information scrolls past with this parameters window.

If you are running KDE as your desktop manager, you could actually have a constant source of information and news as your desktop background itself. Right click on the desktop, and then choose Configure Background. Now, look at that Background tab. You should see a drop-down list, and in that list you should see an option for Background Program.

While you are here, check out the kwebdesktop background program. This (temporarily) brings us back to François' thirst for news or stocks. Click Modify and look at the Command definition:

```
kwebdesktop %x %y %f http://www.kde.org/
```

What this will do is display a specific web page on your desktop. You could decide on a page other than the KDE site. Perhaps a national news web site or a financial information page, *non*? While the KDE page is wonderful to look at, sometimes it may be more valuable to keep track of those volatile investments.

Stocks or emerging markets were not the only thing on François' menu. *Non*, he decided that some internationalization was in order. Since he does not get out much, I am sorry to say, he decided to do his exploring from his Linux desktop, taking advantage of the Mesa 3-D or OpenGL libraries. In fact, the next two items on the menu both require 3-D libraries to be installed. He started with a wonderful visualization tool called Xplanet, created by Hari Nair.

Xplanet is inspired by the xearth screensaver program. It makes use of photo-realistic images that can be obtained from various sources. The package comes with a couple of NASA images that are quite nice, including an “Earth by night” image that is both breathtaking and frightening—breathtaking because it is so beautiful, but frightening to those of us who still like the idea of doing a little stargazing. Speaking of stargazing, the program allows you to map other spherical maps as well. From the Xplanet web site, there are links that will show you where to get different views of this planet and others. Xplanet works in a screensaver mode, as a background for your desktop, and it is dynamic.

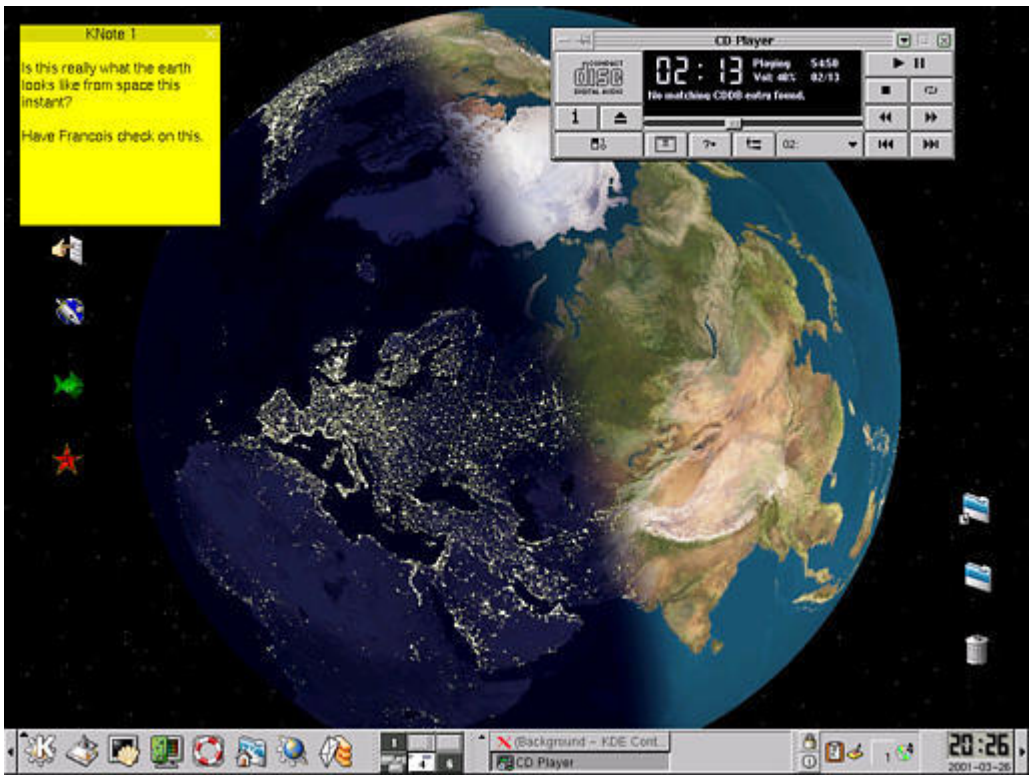


Figure 2. Xplanet

Start by picking up the latest source from the Xplanet site. Then, extract it into a temporary directory:

```
tar -xzvf xplanet-0.80.tar.gz
cd xplanet-0.80
./configure
make
make install
```

What François particularly likes is the fact that you can use Xplanet to explore the planet. For this to work, your system must have the Mesa 3-D libraries installed. If you call up the program in the following manner, you can start on your journey:

```
xplanet -animate
```

You should find yourself looking at a nice rotating globe of the Earth. Using the Home and End keys allows you to zoom in and out. The left and right cursor keys let you play with the rotation of the planet, while the up and down keys change your viewing angle. It's all very cool.

In terms of using the program to explore, Xplanet also comes with a nice Tcl interface called tkxplanet to let you change the look, size, projection and so on, of your world. You can generate the desktop or simply run the projection of your choice in a window.

To run Xplanet as a background, you should be aware of some differences depending on the desktop that you are running. Different desktop managers treat their root windows differently. With Window Maker and GNOME, for instance, all you have to do is decide what kind of a view you want and enter it at the command line—like this:

```
xplanet -moonside
```

This will give you a nice background of the Earth as seen from the Moon. If you try this with KDE 2.1, you might find that nothing happens. That's because the control of the desktop is dealt with somewhat differently, and this is where you define your dynamic background. You might even notice that there is already an entry for Xplanet. I took the original definition and modified it to give me a different view of the world. Here's the program definition from mine:

```
xplanet --geometry %xx%y -observer 49,60 --output %f.jpg && mv %f.jpg %f
```

What happens here is that the Xplanet program generates the map, which is then saved to a background image, which is then written to the desktop. The default refresh rate of ten minutes can be modified to suit your tastes.

For a nice 3-D visualization, we did have an additional option on tonight's menu. Those of you who own your own personal handheld GPS unit (global positioning system) may find this particularly interesting. GPS3D is a system that lets your Linux system communicate with the GPS to pinpoint your location on the planet, superimpose downloaded maps on that same 3-D globe of the Earth, map waypoints and view satellite trajectories. Currently, the program only works with devices that support the NMEA series of protocols (National Marine Electronics Association). To get your copy, head over to http://www.mgix.com/gps3d/. Then, extract and build the program from source (although binaries are available):

```
bunzip2 gps3d-1.16.tar.bz2
tar -xvf gps3d-1.16.tar
cd gps3d-1.16
make
make install
```

The package includes a few programs, one of which is a d<\#230>mon that listens for a serial-connected GPS device and rebroadcasts the information to wherever you want. (Do you want the world to beat a path to your door?) The included program that interests us the most today, however, is called viz, and you can also guess what it does, *non*?

```
./viz &
```

The great thing about this program is it provides some fun for those who do not have their own GPS unit, by displaying a 3-D globe that one can navigate, zoom in and out of and so on. While it sounds like it would be much more fun with a GPS, this is still a fascinating program and a good one to watch (see Figure 3).

Figure 3. GPS3D lets you pinpoint your location on the globe.

Perhaps one of the best ways to get to know how this world works is to try running it yourself, *non*? At least, that is what François thinks, based on the last item on his menu. Before you run out the door and try to have yourself elected dictator somewhere, why not relax and let my ambitious waiter pour you a little bit more wine. I am not talking about controlling the real world, but something better. A world of your own making. A world where you can start from nothing and build entire civilizations.

You may already be familiar with another version of this game. I must tell you right now that *Freeciv* is a different beast and worth your attention. The game can be played across a network with up to 30 different players. If your skills at world conquest are as good as mine, you may want to first play a few games on your own. I tried very hard to build the "Despotism of Canada" and failed on several occasions, even though I populated my capital city of Ottawa with scientists (Figure 4). Imagine.

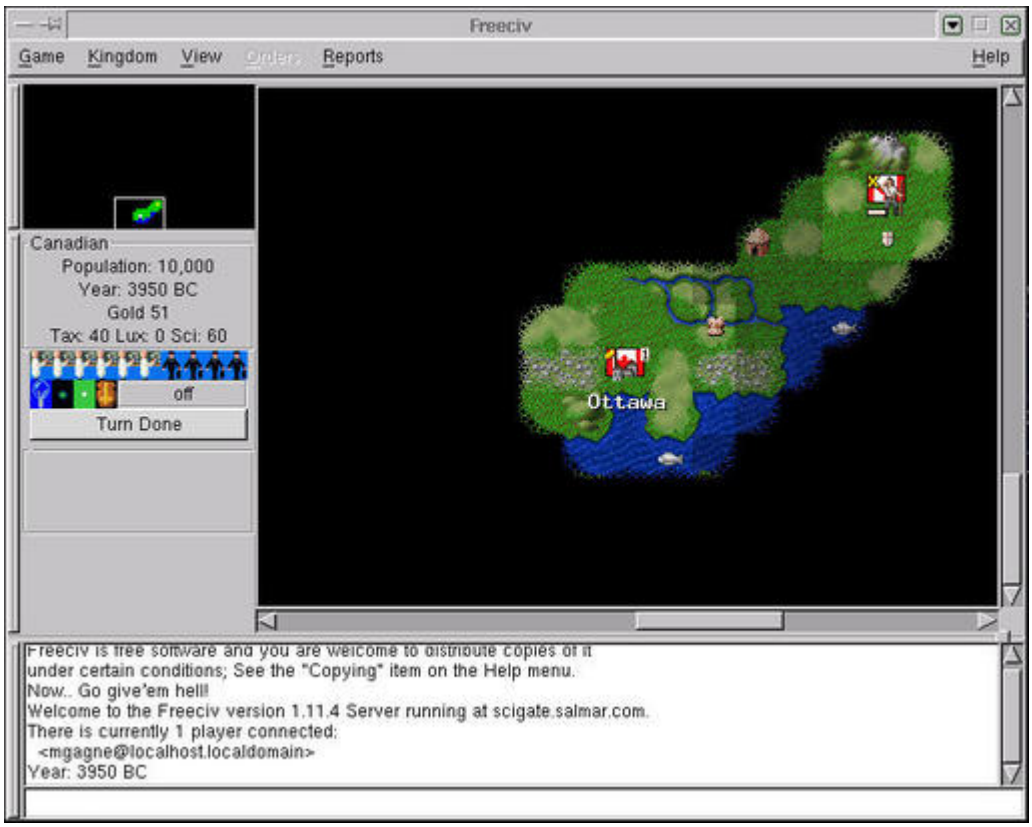


Figure 4. Marcel tries to build the Despotism of Canada.

If you think you can do better than this chef, I invite you download your own copy of *Freeciv* (http://www.freeciv.org/) and begin building your empire by building the code:

```
tar -xzvf freeciv-1.11.4.tar.gz
cd freeciv-1.11.4
./configure
make
make install
```

It is all very easy. While this is all happening, why not try the crème brûlée. It is positively transcendent tonight.

The game consists of both a client (**civclient**) and a server (**civserver**). You start by running a server and waiting for connections. The server itself is a text screen with a command prompt. A number of options can be set here, such as

the size to which a city must grow before the people start becoming unhappy, or for that matter, how much food is required for the city to grow.

Players must start their client by typing **civclient**. The next step is to choose a country and the name of the person who will lead this country. If you have always dreamed of being Hannibal and running the Carthaginian world, or Napoleon in France, then this is your big chance. When you have enough players (one is sufficient), you can start the game by typing **start** at the civserver prompt.

Explore, expand, build, invade. Let Chef Marcel give you a tip though. Too many scientists tend to work out better than too many Elvises. Who would have known?

For those of you who feel a sense of déjà vu, you are no doubt thinking this sounds an awful lot like a complex version of a very old game called *Hammurabi*. For fun (and nostalgia), I decided to see if I could find this ancient wonder of the gaming world. I located it at ftp.sco.com/skunkware/src/games and compiled it on my Linux system. This is a single source file called hammurabi.c, and you can compile it like this:

```
cc hammurabi.c -o hammurabi
```

You can execute the resulting binary by typing **./hammurabi**. The result is something like this:

```
HAMMURABI . . .
    BUY HOW MANY ACRES?20
 *  YOU ARE BUYING 20 ACRES.
    HOW MANY BUSHELS SHALL WE DISTRIBUTE AS FOOD?2000
 *  YOU ARE DISTRIBUTING 2000 BUSHELS.
    HOW MANY ACRES SHALL WE PLANT?1000
--> HAMMURABI! THINK AGAIN -- YOU ONLY HAVE
--> 100 PEOPLE, 1020 ACRES, AND 420 BUSHELS
        IN STOREHOUSES.
    HOW MANY ACRES SHALL WE PLANT?
```

Perhaps I had better stick to the restaurant business, *non*? My attempts at ruling even a make-believe world have been nothing short of disastrous. Still, they say practice makes perfect. Why don't you all log in, *mes amis*, and show me how it is done? And, never fear François, you did a wonderful job with your first menu. Now, refill our guests' glasses one last time before we close for the night.

Mes amis, thank you for coming. Until next time, please join us here at *Chez Marcel*. Your table will be waiting. Of course, I may handle the next menu myself.

A votre santé! Bon appétit!

Resources

Marcel Gagné 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 is coeditor of TransVersions, a science fiction, fantasy and horror anthology. He loves Linux and all flavors of UNIX and will even admit it in public. He is the author of *Linux System Administration: A User's Guide*, coming soon from Addison Wesley Longman. He can be reached via e-mail at mggagne@salmar.com.

Archive Index Issue Table of Contents

Advanced search

Checking Your Work with Scanners, Part II: Nessus

Mick Bauer

Issue #86, June 2001

Take security evaluation and vulnerability reduction to a higher level with Nessus.

Last month we began exploring the dangerous and fun world of scanning, focusing on the powerful port scanner **Nmap**. Nmap helps system administrators and security auditors (and yes, prospective crackers too) determine for what services a given host is accepting connections.

Seeing what points of entry a host offers is a good start in evaluating that host's security. But how do we interpret the information Nmap gives us? For example, in one of the scans we tried last month, the output looked like Listing 1.

Listing 1. Nmap Scan Using TCP Connect, UDP and RPC Modules

Just what does this mean? Sure, we know this host is running a web server (on TCP 80), some level of RPC services (UDP 111, UDP 1026) and probably Windows shares, too (UDP 137, TCP 138-139). But which of these services are actually exploitable?

This is where security scanners come in. At the risk of getting ahead of ourselves, let's look at the output from a **Nessus** scan of our test target (see Figure 1).

Figure 1. Nessus Scan of Windows 98 Host from Listing 1

Space doesn't permit me to show the entire (expanded) report, but even this abbreviated version shows that Nessus identified seven apparent “holes”, or potentially exploitable vulnerabilities in our target system. It also generated four additional warnings and provided two supplemental security notes.

Among other things (you can't see all of this in Figure 1, so you'll have to take my word for it), Nessus determined that this host was running the Sambar web server with no administrative password and with the dangerous **mailit.pl** cgi-script, had its entire C:\ drive shared without any password set (and even if one had been set, Nessus determined that this system was vulnerable to both “Null session” connections and “first-letter” passwords), was running an FTP server on TCP port 1432 (which Nmap had incorrectly guessed was running the blueberry-lm service) and had a TCP/IP stack that used predictable TCP sequence numbers. These can be exploited a number of ways, including TCP-hijacking and IP-spoofing attacks.

Yow! This host is ripe to be owned.

So, what is this deadly magic called Nessus? And why did it dance little circles around Nmap when it came to analyzing this system?

Security Scanners Explained

Whereas a port scanner like Nmap (which is the gold standard in port scanners) tells you what's listening, a security scanner like Nessus tells you what's vulnerable. Since you need to know what's listening before even trying to probe for actual weaknesses, security scanners usually either contain or are linked to port scanners.

As it happens, Nessus invokes Nmap as the initial step in each scan. Thus, it was misleading of me to imply that Nessus out-analyzed Nmap: Nessus depends on Nmap.

Once a security scanner has determined which services are present, it performs various checks to determine which software packages are running, which version each package seems to be at and whether they're subject to any known vulnerabilities. Predictably, this level of intelligence requires a good vulnerability database that must be updated periodically as new vulnerabilities come to light.

Ideally, the database should be user-editable, i.e., it should be possible for you to create custom vulnerability tests particular to your environment and needs. This also ensures that, should the scanner's developer not immediately release an update for a new vulnerability, you can create the update yourself. Not all security scanners have this level of customizability, but Nessus does.

After a security scanner locates, identifies and analyzes the listening-services on each host it's been configured to scan, it creates a report of its findings. The better scanners don't stop at pointing out vulnerabilities; they explain them in detail and suggest how to fix them.

So meaty are the reports generated by good security scanners that highly paid consultants have been known to present them as the primary deliverables of supposedly comprehensive security audits. This is a questionable practice, but it emphasizes the fact that a good scan produces a lot of data.

There are a number of free security scanners available: VLAD, SAINT and Nessus are just a few. Nessus, however, stands out as a viable alternative to powerful commercial products, such as ISS' Internet Scanner and NAI's CyberCop Scanner. Developed primarily by Renaud Deraison and Jordan Hrycaj, Nessus surely ranks with the GIMP and Apache as tools that equal and in many ways exceed the usability and flexibility of their proprietary counterparts.

Before we go any further, I should repeat last month's caution: knowledge is power—use it responsibly! Tools such as Nessus and Nmap should only be run against systems and networks you're authorized to scan. And note that unauthorized port scanning, while frowned upon, is generally not illegal, but unauthorized security-scanning can get you into a great deal of trouble. Consider yourself warned.

Nessus' Architecture

Nessus has two major parts: a server, which runs all scans, and a client, with which you control scans and view reports. This distributed architecture makes Nessus flexible and also allows you to avoid monopolizing your workstation's CPU cycles with scanning activities. It also allows you to mix and match platforms; you can use the UNIX variant of your choice as the server, with your choice of X, Java or MS-Windows clients. (Note, however, the Java client no longer appears to be in active development.)

nessusd listens for client connections on TCP port 3001 and also TCP 1241 (1241 was recently assigned to Nessus by the Internet Assigned Numbers Authority; 3001 will be phased out eventually). Client sessions are authenticated using an El Gamal-based public-key scheme and encrypted using a stream cipher whose session key is negotiated dynamically for each connection. In this regard, Nessus' cipher layer (implemented by Jordan Hrycaj using his libpeks library) behaves similarly to SSL.

Nessus' client component, **nessus**, can be configured to log in either transparently (i.e., with no password associated with your private key) or with a password that protects your private key, thereby preventing unauthorized users from connecting to the Nessus server from your workstation.

Once you've connected with a Nessus server, you're presented with a list of plugins (vulnerability tests) supported by the server and a number of other options. If you've compiled it into Nessus, you may also be given the option to run a detached scan that can continue running even if you close your client-session. A whole page of options pertaining to the creation and maintenance of a knowledge base can also be compiled in, allowing you to store scan data and use it to track your hosts' security from scan to scan (e.g., to run differential scans).

Note that these are both experimental features; they must be explicitly compiled into Nessus due to minor stability issues, but these will have been fixed (and the features fully integrated) by the time Nessus version 1.2 is released. I mention them here because the Detached Scan feature in particular is a good example of the value of Nessus' client-server architecture.

Once you've configured and begun a scan, Nessus invokes each appropriate module and plugin as specified and/or applicable, beginning with an Nmap scan. The results of one plugin test may affect how or even whether subsequent tests are run; Nessus is pretty intelligent that way. When the scan is finished, the results are sent back to the client. If the session-saving feature is enabled, the results may also be stored on the server.

Getting and Installing Nessus

Nessus, like most open-source packages, is available in both source-code and binary distributions. Red Hat 7.0 binaries of Nessus version 1.0.7a (the latest version as of this writing), however, are available only from redhat.aldil.org/rpm.html?id=73, courtesy of Matthias Saou (these binaries have not been compiled with the experimental features).

If you don't use Red Hat 7.0, if your distribution doesn't have its own Nessus packages (Debian 2.2 does) or if you want to use experimental features, you'll need to compile Nessus from source. Not to worry, if you first install a few prerequisites and follow Nessus' installation instructions, this compile should go smoothly. The Nessus FAQ (www.nessus.org/doc/faq.html) and Nessus Mailing List (list.nessus.org) provide ample hints to compile and install Nessus.

Nessus' prerequisites are: Nmap, the port scanner we discussed last month; gtk, the GIMP toolkit, including the packages gtk+, gtk+-devel, glib-devel and XFree86-devel; and the scripting environment m4, or libgmp (whose package is simply called gmp).

Once you've installed these, your distribution may have further prerequisites; I'm aware of two such situations. First, gmp-2.0 is needed for Red Hat 7.0 (which usually includes gmp-3.0 but not 2.0; you'll need to use RPM's --force option if you install 2.0 and 3.0 is already in place). This package is available from www.redhat.com/swr/i686/gmp-2.0.2-5.i686.html.

Second, to install or compile Nessus on SuSE Linux you must first install the packages bison, flex, gtkdev and glibdev. See www.nessus.org/doc/faq.html for more details.

After all prerequisites are in place you're ready to compile and/or install your Nessus packages. Compiling is easy: for each of the four packages simply 1) un-gzip and un-tar it; 2) **cd** into its directory and run **./configure**; 3) **make**; and 4) **make install**. The packages should be compiled and installed in the following order: nessus-libraries, libnasl, nessus-core and nessus-plugins.

Before you run the configure script for nessus-core, consider whether you want to use the session-saving and/or knowledge-base features. Session saving

allows both crash recovery (e.g., the resumption of a scan interrupted by an application or OS crash) and detached scans (see above) and is enabled by including the flag --enable-save-sessions when you run configure.

The knowledge-base feature allows you to store scan results in a database on the server, which in turn allows you to run differential scans. The configure flag to activate the knowledge base is --enable-save-kb.

Thus, if I want to compile Nessus to enable both session-saving and the knowledge base (and their corresponding features), before I compile nessus-core, I'll invoke configure this way:

```
./configure --enable-save-sessions --enable-save-kb
```

See http://www.nessus.org/documentation.html for detailed instructions on compiling and using these features (because they're experimental, that's the last thing I'll say about them in this article).

After all four packages are compiled and installed, make sure that the file /etc/ld.so.conf contains the line /usr/local/lib (if it doesn't, add it with the text editor of your choice). Then, run **ldconfig** to update ld's (the dynamic linker's) cache.

Finally, since one of the strengths of Nessus is the regularity with which the developers add new vulnerability scripts, it makes sense to start out with a complete vulnerability database. If you run the script **nessus-update-plugins**, all plugins created since the current version of Nessus was released will be automatically downloaded to your system using **lynx**. I recommend the usage **nessus-update-plugins -v**, since without the -v flag the script will not print the names of the plugins it's installing. After downloading, uncompressing and saving new scripts, nessus-update-plugins will reset nessusd so it sees the new plugins (assuming a nessus dæmon is active at that moment).

At present this script does not check these scripts against MD5 or other hashes; this mechanism can therefore be subverted in various ways. If that bothers you, you can always download the plugins manually from www.nessus.org/scripts.html, one at a time, but even then you won't know if anything's fishy unless you review each script (they go in /usr/local/lib/nessus/plugins) before the next time you run a scan.

Nessus Clients

Unless you're only going to use the Nessus server as its own client (i.e., run both nessusd and nessus on the same host), you'll need to perform additional installations of Nessus on each host you wish to use as a client. While the Nessus server (the host running nessusd) must be a UNIX host, clients can run

on either UNIX or MS-Windows. Compiling and installing Nessus on UNIX client machines is no different than on servers (as described above).

Installing any of the Windows clients (WinNessus, NessusW and NessusWX) is a bit simpler, as all three are available in binary form. Personally, I prefer WinNessus of the three, since it so closely resembles the UNIX GUI (I'm lazy that way). All three Windows clients are available at www.nessus.org/win32.html.

Before we talk about proper use of the Nessus client, we'd better start our dæmon.

Running and Maintaining nessusd

Okay, we're back at our Nessus server's console and ready to fire Nessus up for the first time. (Are you excited? For good reasons and not evil, I hope!) nessusd is different from many other dæmons in that it can be invoked either as a proper dæmon (i.e, running in the background) or with flags and parameters that reconfigure Nessus. Therefore, to actually start the dæmon in dæmon mode, we enter **nessusd -D**.

As we'd expect with a client/server application, we also need to create some Nessus user accounts on our server. These are independent of the server's local UNIX user accounts. Nessus accounts can be created two different ways. First, we can invoke nessusd with the -P flag immediately followed by a username and one-time password. This neither interferes with a running nessus dæmon nor starts a new one; it does, however, immediately update Nessus' user database and transparently restarts the dæmon.

For example, to add user “bobo” with a password of “scuz00DL”, we enter:

```
nessusd -P bobo,scuz00DL
```

The password is called a one-time password because by default, after bobo first logs in and gives this password, his public key will be registered with the Nessus server; subsequent logins will not require him to enter this password again (they'll be authenticated transparently using an SSL-like challenge/response transaction).

The second and more powerful way to create new user accounts on the server is to use the **nessus-adduser** command. This script actually does most of its magic by invoking nessusd, but it presents you with a convenient interface for managing users with more granularity than a simple nessusd -P. You are prompted not only for a username and one-time password, but also an IP address from which the user may connect and rules that restrict which hosts the user may scan with Nessus.

I leave it to you to read the nessus-adduser man page if you're interested in this level of user account management. Our remaining space here is better spent discussing how to build, run and interpret Nessus scans.

Before we leave the topic of authentication, though, I should mention the other kind of authentication Nessus uses: one local to each client session. When you start Nessus (the client, not the dæmon) for the first time, you are prompted for a passphrase. This passphrase protects a private key stored in the home directory of the UNIX account you're logged into when you start Nessus, and you'll be prompted for it every time. Then, when you connect to a Nessus server, your private key will be used in the transparent challenge/response transaction I described earlier; it actually authenticates you to the remote nessusd process.

If all this seems confusing, don't worry: just remember that the password you're prompted for each time you start Nessus has nothing to do with the password you use the first time you connect to a Nessus server.

Performing Security Scans with Nessus

Now the real fun begins! After Nessus has been installed and at least one user account set up, you're ready to scan. First, start a client session and enter your client-private-key's passphrase when prompted (by the way, you can change or delete this passphrase with the command **nessus -C**, which will prompt you for your current passphrase and what you'd like to change it to).

Next, enter the name or IP address of the Nessusd host (server) you wish to connect to, the port it's listening on, your preferred encryption method and your Nessus login/username (Figure 2). The defaults for Port and Encryption are usually fine.

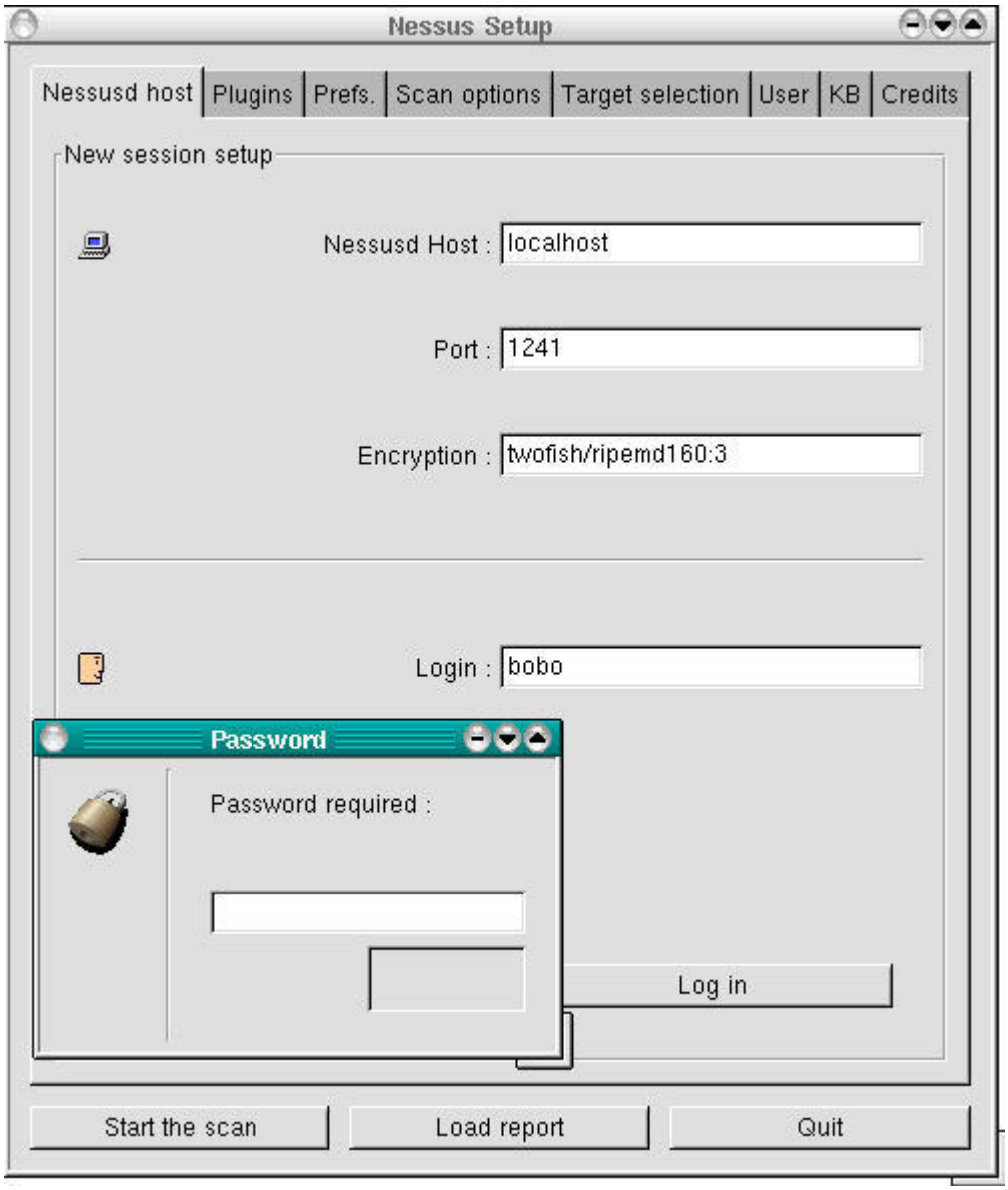


Figure 2. User “bobo's” First Login to a Nessus Server

When you're ready to connect, click the Log in button. If this is the first time you've connected to the server using the specified login, you'll be prompted for your one-time password (next time, you won't be). And with that, you should be connected and ready to build a scan.

If you click the Plugins tab, you're presented with a list of all vulnerability tests available on the Nessus server, grouped by family (Figure 3). Click on a family's name (these are listed in the upper half of the window) to see a list of that family's plugins. Clicking on a family's check-box allows you to enable or disable all its plugins.

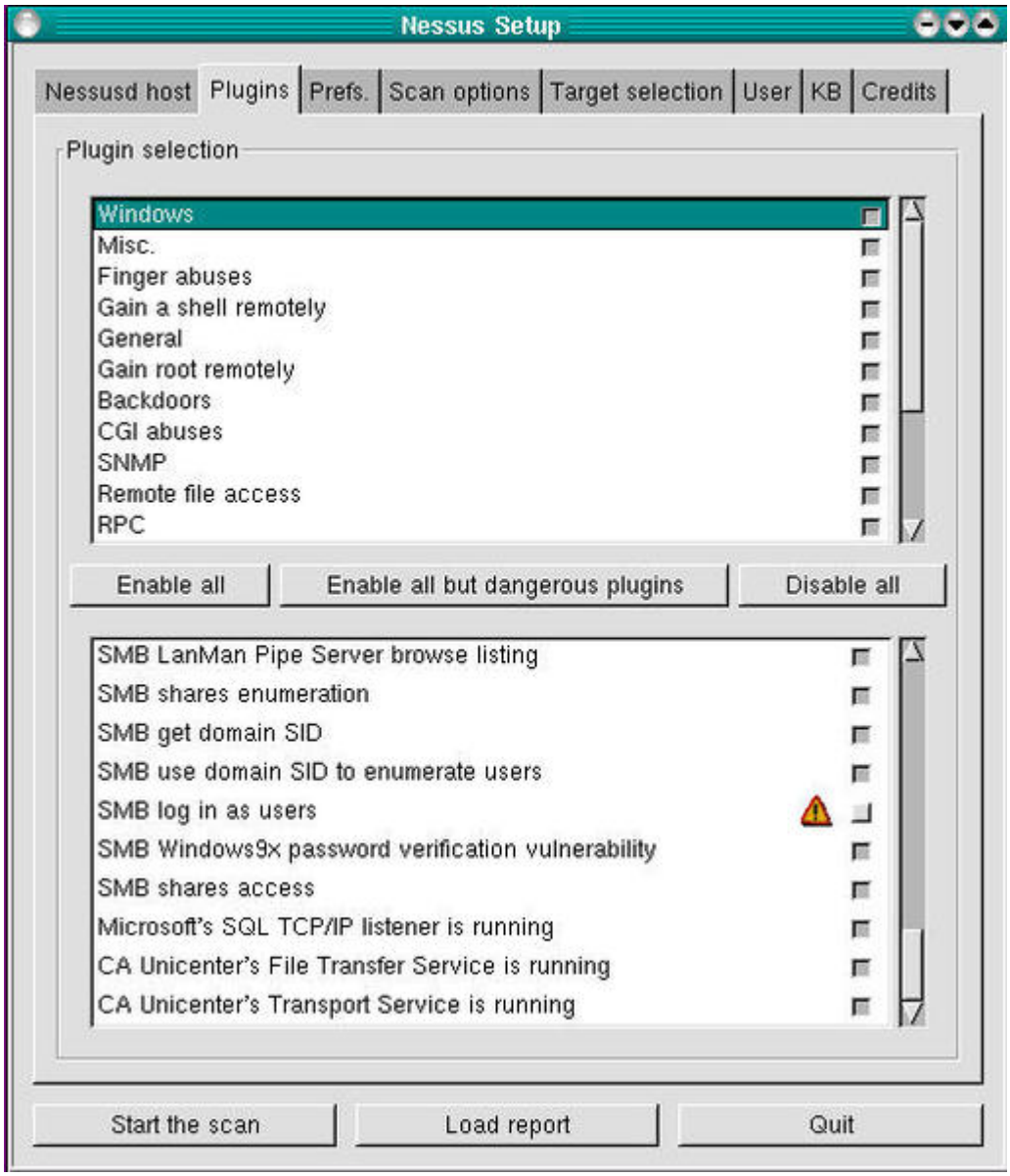


Figure 3. Plugins Screen (Windows Family Selected and Displayed)

If you don't know what a given plugin does, click its name and an information window will pop up. If you hover the mouse-pointer over a plugin's name, a summary caption will pop up that briefly states what the plugin does. Plugins with yellow triangles next to their check-boxes are dangerous: the particular tests they perform have the potential to interrupt or even crash services on the target (victim) host. Enable these with extreme care.

Don't be too worried about selecting all, or a large number of, plugins. Nessus is intelligent enough to skip, for example, Windows tests on non-Windows hosts. In general, Nessus is efficient in deciding what tests to run and in what circumstances.

The next screen to configure is Prefs (Figure 4). Contrary to what you might think, this screen contains not general, but plugin-specific, preferences, some of

which are mandatory for the corresponding plugin to work properly. Be sure to scroll down the entire list and provide as much information as you can.

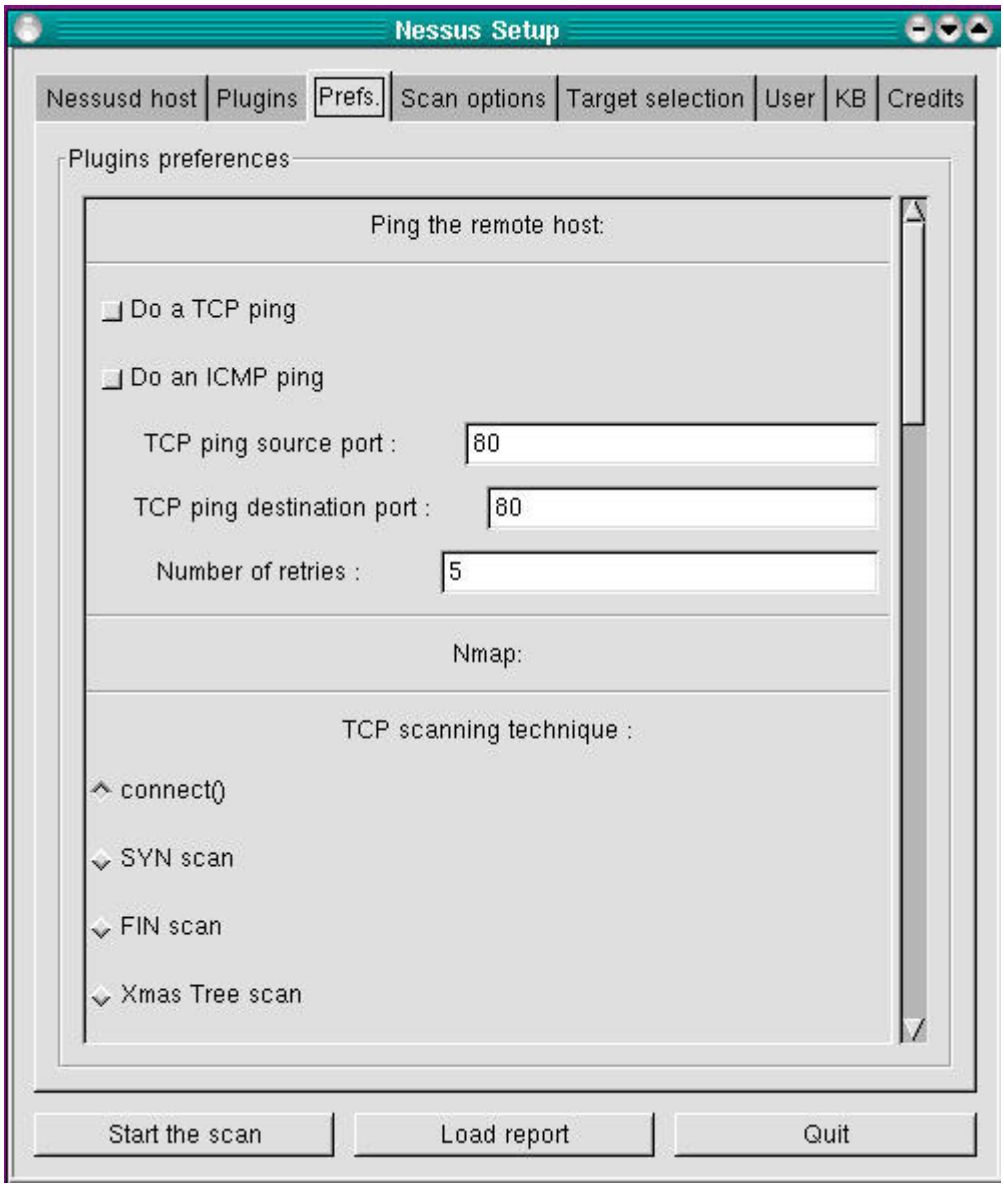


Figure 4. Preferences Screen

Take care with the Ping section (at the very top); more often than not, selecting either ping method (TCP or ICMP) can cause Nessus to mistakenly decide that hosts are down when in fact they are up. Nessus will not perform any tests on a host that doesn't reply to pings—when in doubt, don't ping. Warning: in the Nmap section, Linux users should select only tcp connect() and should deselect all other scan types, due to a bug in libpcap that affects the way Nessus performs port scans.

After Prefs comes Scan Options (Figure 5). Note that the Nessus installation in Figure 5 was compiled with the save-session feature, as evidenced by the Detached Scan and Continuous Scan options, which would otherwise be

absent. As in the Prefs screen, you should deselect everything under Port scanner except Nmap tcp connect() scan due to the bug mentioned above.

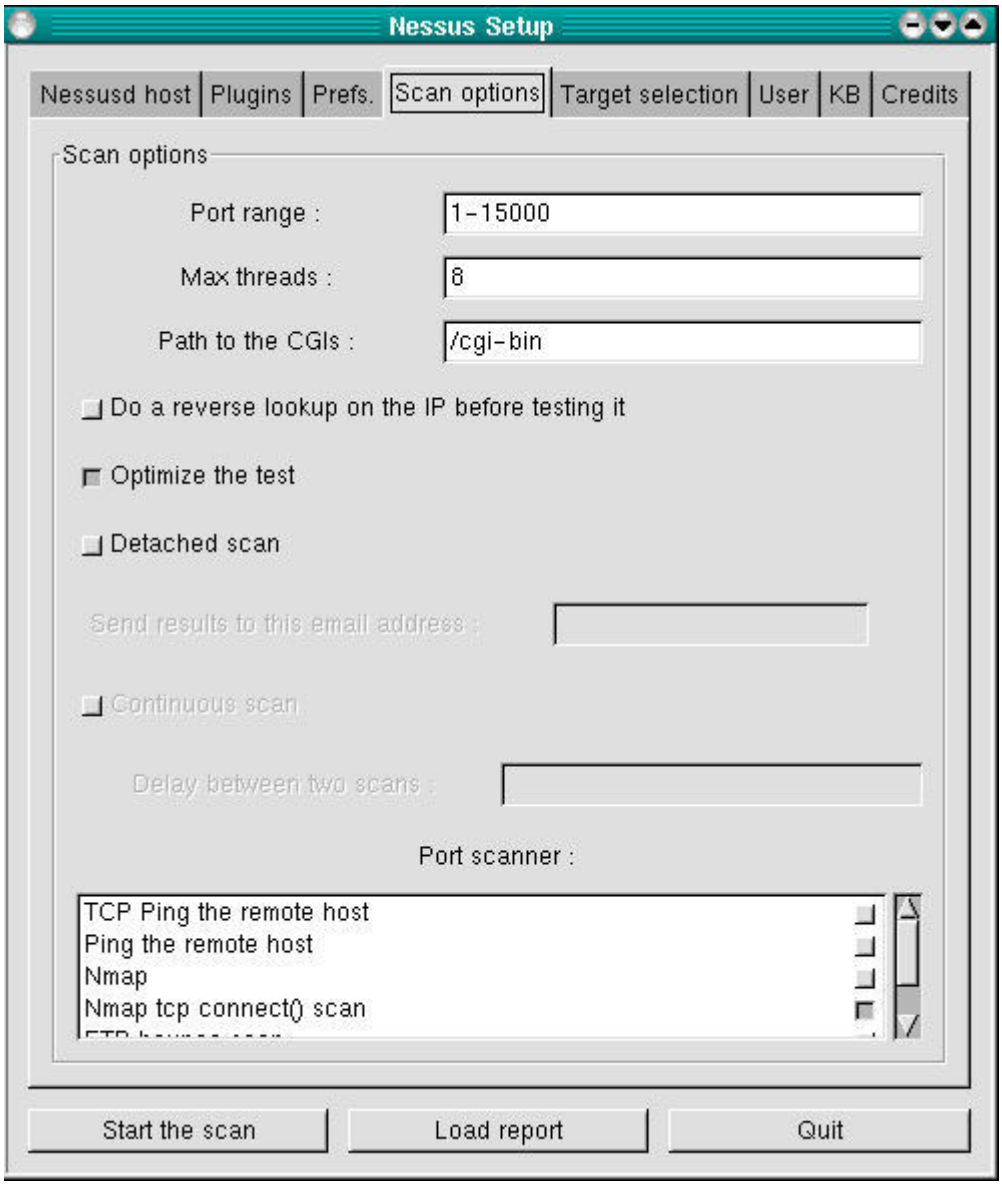


Figure 5. Scan Options Screen

The “Optimize the test” option tells Nessus to avoid all apparently inapplicable tests, but this can result in false negatives, at least theoretically. Think about how much that worries you versus whether you want the scan to complete as quickly as possible. Speaking of speed, if you care about it you probably want to avoid using the “Do a reverse (DNS) lookup...” feature; it attempts to determine the hostnames for all scanned IP addresses.

And now it's time to specify our victims...oops, I mean targets. We specify these in the Target(s): field of the Target Selection screen (Figure 6). This can contain hostnames, IP addresses and network addresses in the format *x.x.x.x/y* (where *x.x.x.x* is the network number and *y* is the number of bits in the subnet mask, e.g., 192.168.1.0/24) in a comma-separated list.

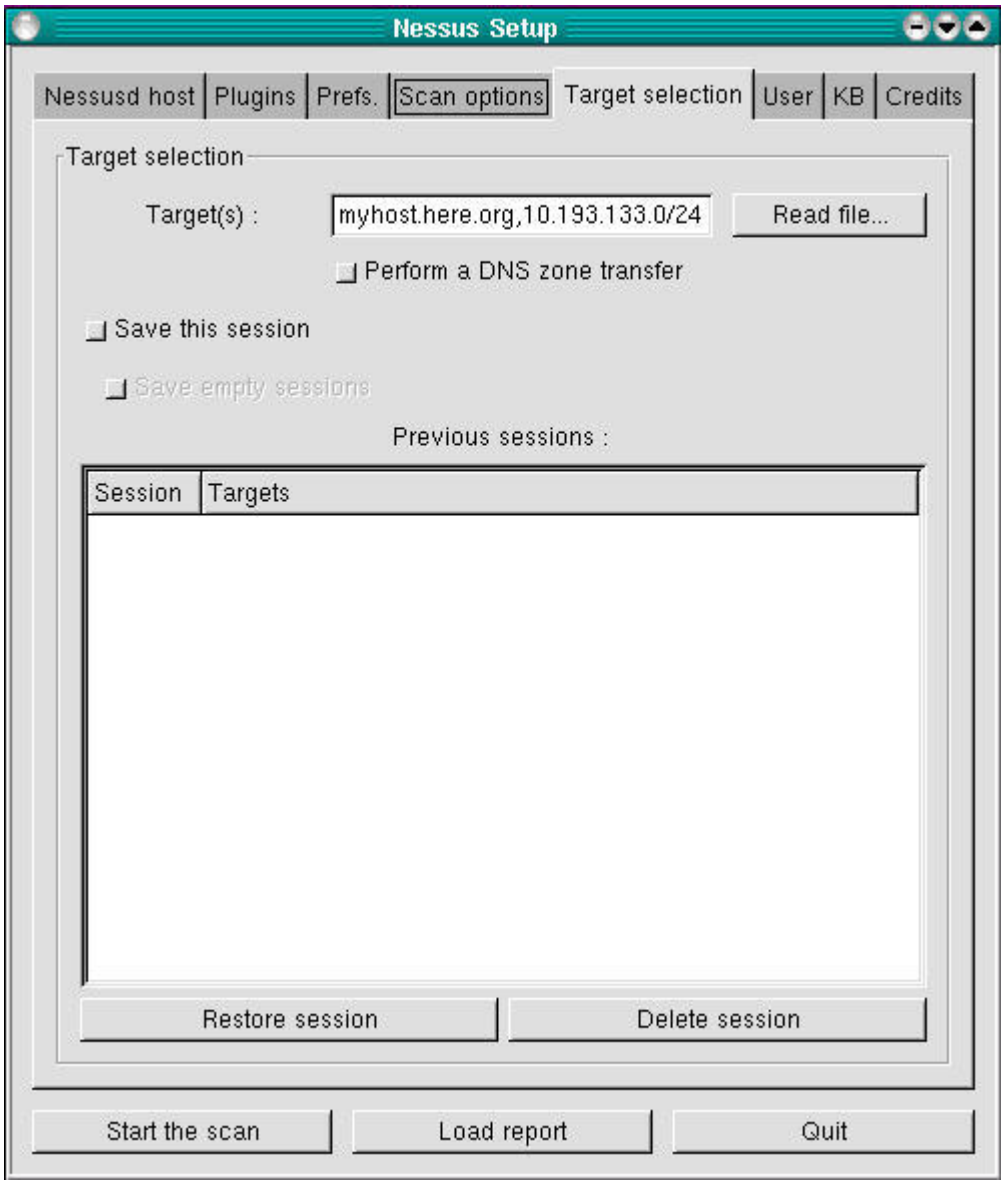


Figure 6. Target Selection Screen

The “Perform a DNS zone transfer” option instructs Nessus to attempt to obtain all available DNS information on any domain names or subdomain names referred to in the Target(s): box. Note, most internet DNS servers are configured to deny zone-transfer requests from unknown hosts. The other options in this screen have to do with the experimental save-session feature I mentioned earlier—see www.nessus.org/documentation.html for more information on what the experimental features do and how to use them.

Finally, one last screen before we begin our scan: User (see Figure 7). (We're skipping KB, which only applies if you've compiled and wish to use the knowledge-base features.) In this screen, we can change our client passphrase (this has the same effect as nessus -C), and we can list exceptions (fine tunings, really) of the targets we specified in the Target Selection screen.

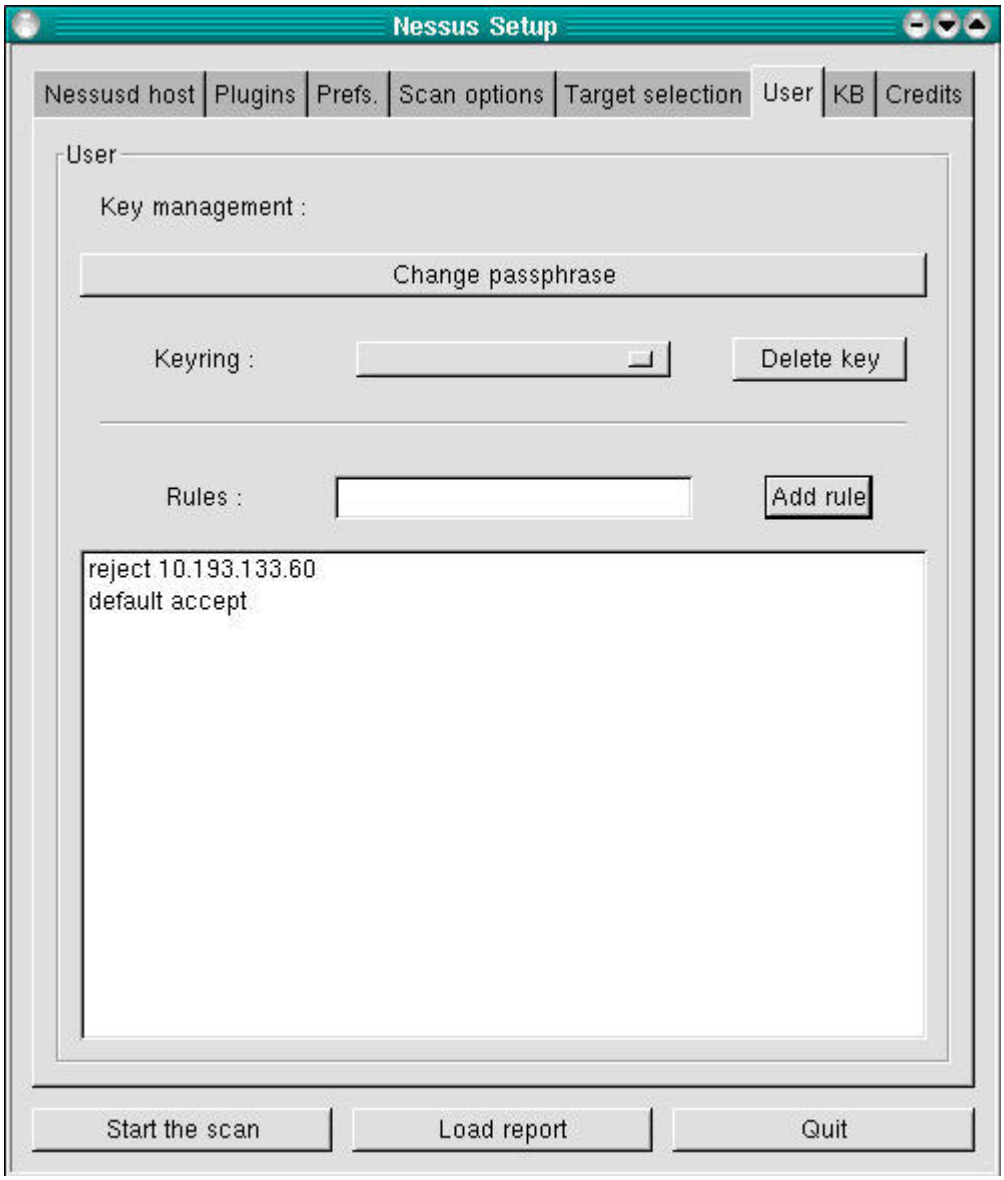


Figure 7. User Screen

These exceptions are called rules, and they follow a simple format: accept address, deny address, default accept or reject. In Figure 7, the rules listed mean don't scan 10.193.133.60, but scan everything else listed in the previous screen.

And now, the payoff. We click the Start the Scan button at the bottom of the screen, and we're off. The scan's length will vary, depending mainly on how many hosts you're scanning and how many tests you've enabled. The end result? A report such as that shown in Figure 1.

From the Report window, besides viewing the report and drilling down into its various details, you can save the report to a file. Supported report file formats include HTML, ASCII, LaTeX and, of course, a proprietary Nessus Report format, NSR (which you should use for reports you wish to view again within Nessus).

Read this report carefully; be sure to expand all + boxes and fix the things Nessus turns up. Nessus can find problems and can even suggest solutions, but it won't fix things for you. Also, Nessus won't necessarily find everything wrong with your system.

It's a fact of life with security scanners; they can only do so much, and not all plugins are equally effective at finding the things they're supposed to find. Even to the extent they are effective, Nessus obviously can't find vulnerabilities it doesn't have plugins for, so be sure to update your plugins regularly.

Some Parting Thoughts

Nessus is a powerful, flexible, commercial-grade, but completely free security scanner. When generated and interpreted properly, Nessus reports can help you stay ahead of the well known vulnerability curve. We haven't discussed how to write custom plugins, but these allow you to test not only for commonly known vulnerabilities but even brand-new or hitherto unknown exploits.

Again, please, use this tool responsibly. Assuming you will, have fun!

Mick Bauer (mick@visi.com) is a network security consultant in the Twin Cities area. He's been a Linux devotee since 1995 and an OpenBSD zealot since 1997, taking particular pleasure in getting these cutting-edge operating systems to run on obsolete junk. Mick welcomes questions, comments and greetings.

Archive Index Issue Table of Contents

Advanced search

AVI Movie Players and Capture

Robin Rowe

Issue #86, June 2001

Robin continues to test video and audio players for Linux.

Last month we tried some MPEG players, including aKtion, gmpeg, gxanim, MPlayer, plaympeg, XAnim, xine and Xtheater. We also looked at the Be operating system in order to compare its video capabilities with Linux.

Our goal this month is to look at the AVI players aKtion, Aviplay, MPlayer and XAnim. We will also capture some AVI files using Video4Linux and XawTV or QtVidcap.

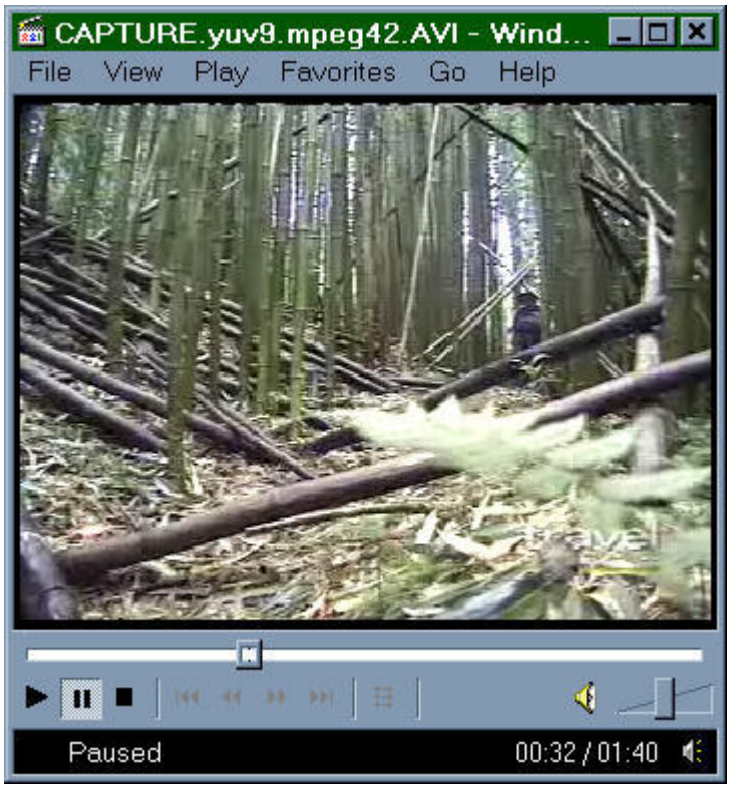
AVI (audio video interleave) is one of the most common formats for video files on PCs. The AVI file format and the RIFF format (resource interchange file format) generally, were defined by Microsoft and IBM back when they were still friends. The Microsoft audio WAV format is another variant of a RIFF file. The RIFF format was based upon IFF, a format standard on the Amiga. When seeking information about the AVI format, a good place to start is John McGowan's “AVI Overview”, available at www.jmcgowan.com/avi.html or www.faqs.org.

Internally, video files are chunks of interleaved pictures and audio. To play a video, those chunks are read sequentially one after the other by a video player and mixed together in a way that makes them appear seamless. Typically, audio playback is the reference and the video picture is synchronized to it. Variations in audio playback rate would be discernible as wow and flutter, but slight variations in video playback framerate are not apparent.

Video file formats rely upon two easily confused concepts: transports and codecs. A transport is the structure that describes the chunks of audio and video that follow. AVI, MPEG and QuickTime are all examples of transports. Within the transport, each compressed chunk of video or audio must be read using a codec that understands the particular scheme of that chunk. Examples

of video codecs include Cinepak and Indeo. AVI sound codecs may be PCM, ADPCM, GSM or many others. If you have what seems to be a perfectly good video file, but your video player only plays the video or audio part, it is probably because you are missing a codec. Players are designed to skip over things in the transport they don't understand.

WMP (Windows Media Player) is the movie player included with Windows. It uses installable codecs that work as plugins, called dlls (dynamic link libraries). If you develop a new Windows codec, your video or audio can play back in WMP. For Linux codec developers, this means it is feasible to port your code to Windows as a plugin, so that files created with open formats in Linux can be viewed in the proprietary Windows player. Conversely, some Linux video players are able to use Windows plugins and will play proprietary Windows formats in otherwise open Linux players. It is therefore possible to deliver a single plugin that will work for both Linux and Windows, at least on Intel.



The Windows Media Player

AVI files identify their embedded codecs using a magic number called a FOURCC code. This four-byte id is how the AVI transport knows what codec to load in order to play a stream. To guarantee uniqueness, the FOURCC must be registered with Microsoft. The unofficial but “Almost Definitive” FOURCC Definition list is kept at http://www.webartz.com/fourcc/ and provides a lot of good information.

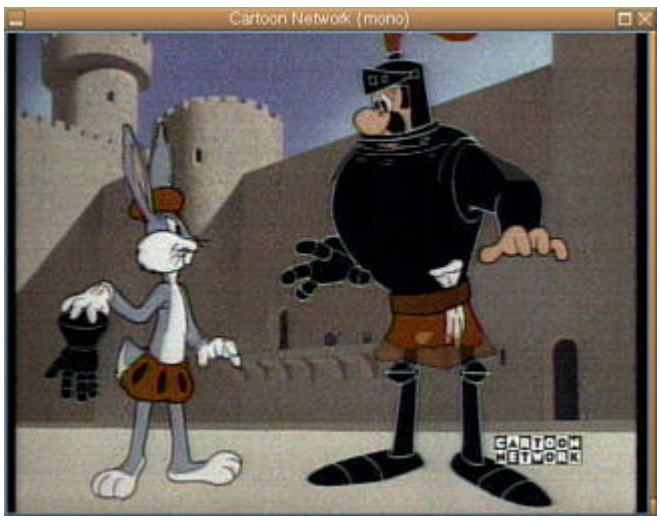
It is widely believed that AVI files can't be larger than 2GB or used for professional video applications, but that's not really true. The main reason for the file-size limitation is that a RIFF block uses a 32-bit integer to describe its size. You may hear this alternately referred to as a 1GB limit because a bug in the early Microsoft Video for Windows code made that (huge at the time) the limit. Because an unsigned 32-bit integer can also contain 4GB, you may see that number suggested as a maximum, too—4GB also happens to be the limit for the FAT16 filesystem.

The original AVI file spec only permits one RIFF chunk per file. The obvious fix to the size limit was to allow multiple RIFFs in one file. At the urging of Matrox, that's what the Open Digital Media (OpenDML) Consortium did when it created the OpenDML AVI file format extensions. The spec is available from a link in McGowan's AVI Overview. OpenDML AVI support has been reportedly contained in Matrox's DigiSuite software and in Microsoft WMP since version 5.1.

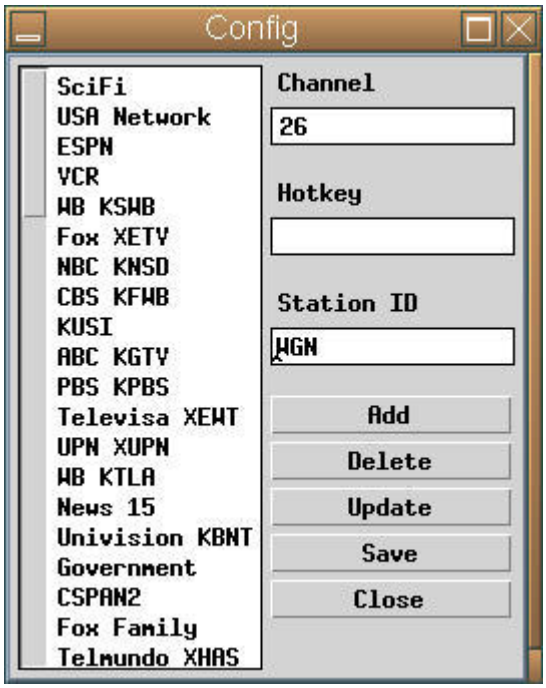
Recently, AVI is losing steam to new formats available from Microsoft; advanced streaming format (ASF) and later Windows Media Format (WMF) were added in WMP 7. Microsoft has tested WMF files as large as 30GB, but they can theoretically hold 17 million terabytes. For audio files, the file extension is wma, for video, wmv.

Some Video Players for Linux

XawTV is a popular application for watching video on Video4Linux devices. Video4Linux (V4L) enables many different video capture cards to be used interchangeably in applications. Hauppauge WinTV is one of the most popular inexpensive TV cards (less than $70 US). Other comparable cards are based on the same Conexant chipset (formerly Rockwell Brooktree).



The XawTV Video Player



The XawTV Config Screen

Ralph and Marcus Metzler created the first bttv Linux driver code to work with Brooktree BT848-based TV cards. Gerd Knorr eventually took over development, creating the bttv-0.7.x series drivers. He also created and continues to maintain XawTV. Alan Cox developed a generic video architecture called Video4Linux that was influenced by ideas in the OSS sound drivers. He developed V4L around the bttv driver and brought it into the Linux kernel. The original bttv driver is no longer supported and has been replaced with the V4L version maintained by Justin Schoeman.

V4L2 is a substantial rewrite of the bttv driver for Linux. According to Justin Schoeman's web page (http://sourceforge.net/projects/bttv-v4l2/), the aim of bttv-v4l2 is to support high performance video capture on Bt848/878-based video capture cards. Bill Dirks is responsible for the current development of the V4L version 2 architecture (V4L2). He developed the capture API and made architectural changes to better accommodate a variety of hardware devices. His web page, http://www.thedirks.org/v4l2/, has more information.

When building and installing the bttv driver (as we did in this column in the April 2001 issue of *LJ*), we encountered something called the i2c driver. This perplexed us at the time because we couldn't get it to work, and none of the V4L web pages say what it does. Looking more closely at Gerd Knorr's web site, we noticed that we needed a 2.3 or newer kernel for i2c and therefore had to do a minor patch to Debian Potato's 2.2 kernel. We had the fix but didn't understand it.

Talking with Bill Dirks finally solved the mystery for us of what the i2c driver does. TV cards have tuners, and they have their own communications bus

called i2c (or iic), defined by manufacturer Phillips. It is a three-wire synchronous serial bus (clock, data, ground) that communicates between the tuner and other components on the card. It is the PCI bus that talks to the i2c. Another place an i2c bus is used is on the PC motherboard so it can talk to the temperature sensor for the CPU.

Although it is still experimental, Bill Dirks recommends using V4L2 unless doing so causes a problem. V4L is more bttv-centric, but apps may use either Video4Linux version transparently. The videodev driver loads the device specific driver in V4L. In V4L2, the driver loader may be videodev or videodevx. The V4L2 videodev package works only with 2.2.x kernels and supports only V4L2 drivers. The new V4L2 videodevx driver works with 2.2.x and 2.4.x kernels and supports both V4L2 and V4L device drivers.

V4L is included with the kernel. To get V4L2, you need to install something extra. Because the version of V4L with Potato is so old, we did that anyway. However, we finally went with V4L because we were scared off by the newness of V4L2. We haven't followed Bill's advice yet to switch to V4L2, which may have something to do with some difficulties we had testing video capture.

We built the AVI players from source, either because we wanted the latest version or because no deb package was available. The procedure is to search on freshmeat.net for the project, use **tar xvfz** or **xvfl** to unpack it (depending on whether the file is gzipped or bzipped), then build it. Building is typically done the GNU way: **./configure**, then **make** and **make install**.

MPlayer was deemed the most stable MPEG player in our evaluation last month, although the 0.11 version we had wouldn't play AVI files. The new version does. It does a pretty good job, but it is obvious that AVI support isn't as mature. Some files didn't play, some had no video and one even played upside down.

In theory, MPlayer can play any video that Microsoft WMP can because it tries to use WMP codec plugins copied into the Linux /usr/lib/win32 directory. However, tricking the WMP plugins to work in Linux isn't entirely perfected yet, and MPlayer continues to suffer from a spartan command-line control interface.

You can't resize the player window, which will annoy most people accustomed to viewing small videos at double size. You also can't get a video to repeat, and quitting doesn't seem to respond very fast.

There is no version number for MPlayer anymore; you can only download a daily CVS snapshot. We had serious trouble getting that to build due to a

problem with our gcc compiler. We received a somewhat cryptic message from gcc about an install problem, and it couldn't execute cc1plus, the g++ front end. However, g++ seemed to be installed correctly. The problem went away when we installed a newer version of gcc (2.95.3-5).

XAnim is popular for viewing both MPEGs and AVIs. It uses plugins but not the Win32 dll ones. It offers its own cross-platform plugins for H.261, H.263, Indeo and Cinepak. Although not many are offered, the docs say the plugins run on Intel, Alpha, PowerPC, Sun and SGI hardware. According to the XAnim web page, the plugins are proprietary, developed under NDA.

The XAnim Player

XAnim worked okay but couldn't play as many AVIs as MPlayer. However, the user interface in XAnim is better. We wanted to try the latest version of XAnim, but we had trouble building it. Building XAnim requires first installing the Imake generator **xmkmf**. It's supposed to be included with the X Window System development libraries, but we couldn't find that. Later we noticed that the MPlayer web page says those libraries are called lib6g-dev in Debian.

The XAnim User Interface

Two other players we didn't get built are aKtion and Kmpg. We are running an older version of aKtion, but it has some problems playing AVI files. We couldn't build Kmpg 0.5.4 because it didn't like our system's too-new version of libqt (2.2.4 instead of 1.x). Not having KDE headers was the build error reported by aKtion 0.4.1. Xtheater 0.9.1 built but wouldn't play any AVIs.

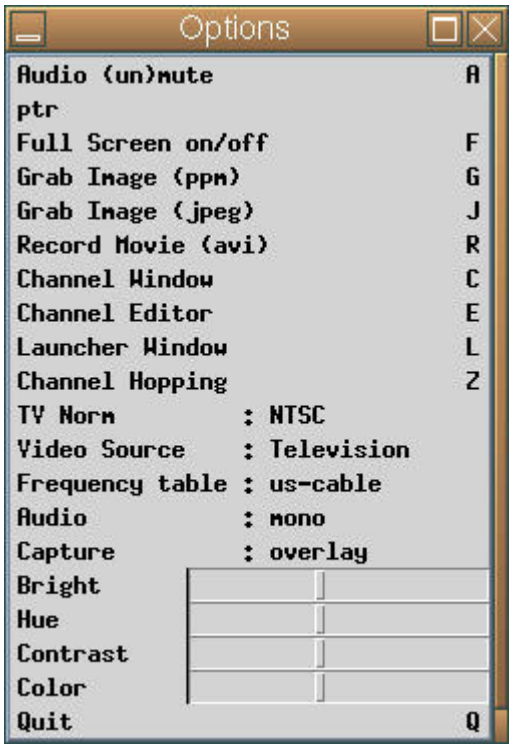
Avifile 0.53.5 is a library for playing AVI files. It comes with two sample applications, Aviplay and QtVidcap. Using Aviplay offers a somewhat different interface than XAnim but is a similar experience. XAnim doesn't seem to use Avifile. MPlayer doesn't use Avifile but they do have the Win32 DLL loader in common. Both seem a bit more reliable than Aviplay.

Aviplay

The QtVidcap capture program looks enticing, but we couldn't get it to work. When it started to capture a file it would fail. We hope it will do better after we install XFree86 4.x and V4L2.

XawTV succeeded in capturing AVI files, but its power seemed feeble compared to the same task in Windows with Windows Media Encoder. XawTV only encodes 15-bit and 24-bit RGB or MJPEG codecs. XAnim was the only player that could play all three XawTV formats. The other players tended to balk, except for MJPEG. Aviplay didn't like any of them. We had trouble playing the files in Windows and wondered if the fault was in the capture. The Linux players did better with Windows-captured video, provided that the codec wasn't too new.

The QtVidcap Program



The XawTV Options Window

Two things puzzled us when operating XawTV: adjusting picture size and sound. You must size the window to the size you want to capture videos. Anything larger than 320 x 240 pixels at 15fps gave us problems. We had no sound until it dawned on us to select “line input for record” in gmixer.

Next month we will upgrade our Debian Linux installation from Potato to Woody, giving us the more multimedia capable 2.4 kernel and XFree86 4.0 GUI. We will also upgrade our device drivers to V4L2.

Robin Rowe is a partner in MovieEditor.com, a technology company that creates Internet and broadcast video applications. He has written for *Dr. Dobb's Journal*, the *C++ Report*, the *C/C++ Users Journal* and *Data Based Advisor*. His software designs include a client/server video editing system in use at a Manhattan 24-hour broadcast television news station, Time Warner New York One and associated web site http://www.ny1.com/. You can reach him at robin.rowe@movieeditor.com.

Archive Index Issue Table of Contents

Advanced search

Downturn Has Silver Lining

Linley Gwennap

Issue #86, June 2001

Falling component prices and decreasing demand will benefit consumers.

The downturn in the economy has eroded demand for PCs and other high-tech gadgets. This erosion has hurt revenues and earnings at most semiconductor companies, but their pain will ultimately be your gain. Falling prices will spur demand for webpads, hard-disk video recorders, digital cameras, MP3 players and similar devices, many of which use Linux.

The chip industry has gone through several boom and bust cycles, and the current situation is just another down cycle. The problem is that building a new chip factory, or fab, costs upward of $1 billion and takes about two years from site selection to volume production.

During periods of high demand, such as last year, semiconductor companies can't add capacity quickly enough. This results in shortages and high component prices. With cash in hand, the companies madly invest in new capacity, but today we see that increasing capacity meeting falling demand.

Basic economics tells us that prices will fall. Indeed, DRAM prices have dropped by 50% over the past six months, as have the prices of flat-panel (LCD) displays. The prices of Flash memory and other components are also sagging. I wouldn't be surprised if, by the end of this year, some components cost a quarter of what they did last year.

These lower prices will benefit many emerging products. For example, webpads have yet to get off the ground. Only a few companies are selling them and generally at prices in excess of $1,000 US. The culprit is the cost of the flat-panel display, which has been several hundred dollars.

With display prices collapsing, vendors will be able to deploy webpads for much less than $1,000, perhaps as low as $500. I have no doubt an early-adopter

market will spring up at these prices, and that market will grow as prices continue to drop. Most of the webpads announced to date combine Transmeta's Crusoe chip with Linux, delivering web compatibility without Windows.

TiVo's ground-breaking video recorder, which stores TV programs on a hard disk instead of videotape, has received critical acclaim but has only sold about 200,000 units. These low sales are due in part to the high cost of the unit, $500 or more (including the cost of the service). When TiVo launched last year, the price of a 30GB hard drive was about $300. Today, the same hard drive costs less than $100. The price of DRAM and other components in the system has fallen as well. Watch for TiVo to take advantage of these changes by dropping the price of its Linux-based recorder. With lower prices, more people will buy the TiVo system to take control of when they watch their favorite TV shows.

Like the TiVo system, an MP3 jukebox is primarily a simple computer wrapped around a large hard drive. These jukeboxes connect directly to home stereo systems and are just becoming available. But prices, such as $799 for the AudioReQuest, have inhibited demand. Falling component prices, however, should make these systems more popular. SonicBlue (which produces the Rio line of MP3 systems), Kerbango and others all use Linux to power their MP3 jukeboxes.

The bulk of the cost of a portable MP3 player is its Flash memory. The 64MB of Flash found in a $250 player cost about $120 at last year's prices. Even that amount of memory holds only 1-2 hours of music (depending on the quality and file type), barely enough to be useful. With lower Flash prices, we'll see low-end players approaching $100 and players with 128MB or more of memory selling for less than $300.

These MP3 systems will continue to grow in popularity despite the demise of Napster because they allow users to digitize their music collection and take it with them wherever they go. Users can easily find favorite songs, delete unwanted songs and create custom playlists. Linux can play a key role in networking a user's MP3 systems to enable seamless music portability.

Unfortunately, falling chip prices won't help PC sales. At $399, PCs are already as inexpensive as they can be without removing key components. Therefore, PC makers will take advantage of lower chip costs to offer faster processors, more DRAM and more disk capacity at the same price.

Some people may be attracted to better PCs at the same price, but a bigger problem will be convincing most PC users they need a better system. Most current PCs are fine for e-mail, web browsing and word processing. But the

increasing popularity of digital video, digital photos and digital audio will drive the next PC upgrade cycle.

Despite the gloom in the industry, technology has not suddenly become a bad idea. The Internet continues to fundamentally change everything we do, and the trend toward digital media is revolutionizing personal entertainment. With chip prices down 50% to 75%, these trends will pick up speed.

Linley Gwennap (linleyg@linleygroup.com) is the founder and principal analyst of The Linley Group (http://www.linleygroup.com/), a technology analysis firm in Mt. View, California.

Archive Index Issue Table of Contents

Advanced search

Spotlight on Embedded Linux at CeBIT

Rick Lehrbaum

Issue #86, June 2001

Embedded Linux shows its stuff!

Embedded Linux was an active player at this year's European technology mega-expo held in Hannover, Germany. Lots of new devices with “Linux inside” were demonstrated or announced at CeBIT.

Sharp finally unveiled their plans for a Linux-based PDA at a CeBIT news conference. It's called the MultiMedia Tool and has a 320 x 240 color LCD and can be used to surf the Web, check e-mail, play MP3 and MPEG4 files, and also comes with a suite of PDA PIM applications. Other features are a pullout keyboard, 32MB of memory, and interfaces to USB and infrared. It also has a Secure Digital slot, the same technology used by Palm. Special software called PacketPlayer supposedly provides efficient streaming of live video from wireless sources. Rumor has it that Sharp received development support from AXE on the GUI and from Red Hat on the embedded Linux OS.

Galleo announced a Linux-based PDA/cell-phone combo device called the Mobile Multimedia Communicator. Like several other recently introduced handheld computers with wireless connectivity, this gizmo combines the functions of a PDA, web appliance and cellular phone. It sports a 320 x 240 TFT LCD, which is used in portrait mode and includes special page software that condenses standard web pages to make them easy to view on the quarter-VGA-sized screen.

Galleo's Linux-based mobile gadget combines PDA, web appliance and cellular functions.

IBM showed off a second-generation Linux wristwatch at CeBIT. This one is smaller than the original and features a bright yellow OLED (organic light-emitting diode) 640 x 480 pixel display. The watch face is now 0.65" x 0.87" in size, and the battery life has been increased to between four and six hours, as a result of some tweaking of the kernel by IBM. Let's see...that means the watch only needs to have six hour marks, instead of the usual twelve, right?

IBM's latest Linux wristwatch boasts a bright 640 x 480 pixel display.

Korean electronics powerhouse LG Electronics has announced a new Linux-based wireless webpad aimed at home users and offering internet connectivity and a built-in multimedia player. LG's Digital iPad is powered by a 206MHz Intel SA-1110 system-on-chip processor with 64MB of SDRAM memory, and it connects to the Internet via an 802.11 wireless transceiver. I/O and memory can be expanded using PCMCIA cards.

Korea's LG Electronics will soon enter the webpad market with this Linux-based device.

Similarly, Taiwan manufacturing giant MiTAC also unveiled a Linux-based, Bluetooth-connected handheld computer called the CAT. It has a 4.1" 320 x 240 pixel backlit monochrome LCD and is amply endowed with interconnection ports, including Bluetooth, IrDA, RS-232 and USB. In addition to all the usual PIM applications and internet support, software is included for playing MP3s and voice record/playback.

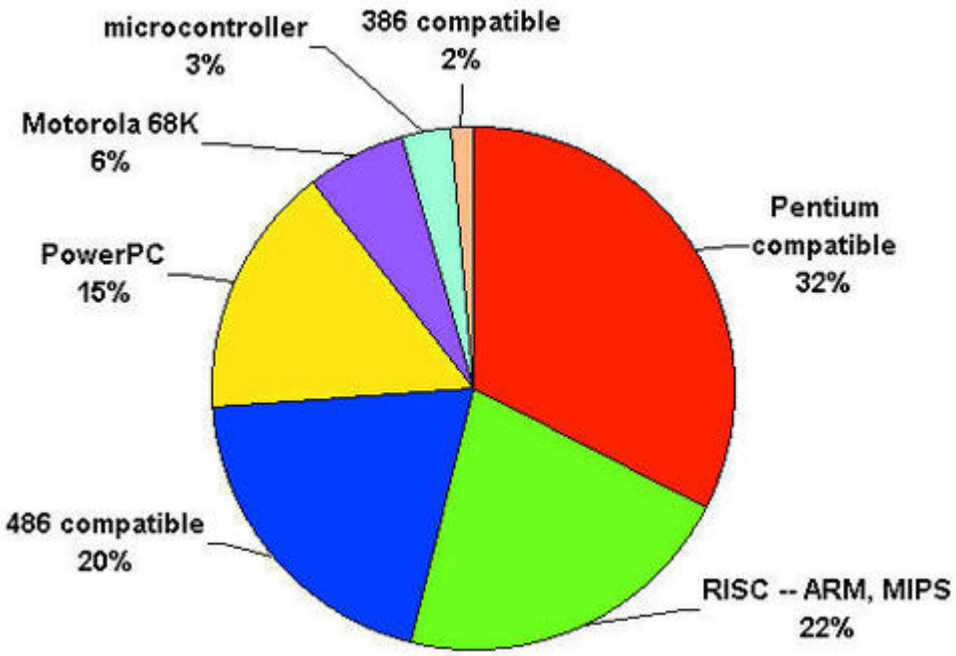
Taiwan's MiTAC is preparing to roll out this Bluetooth-connected handheld Linux computer.

You can now buy iPAQ PDAs preloaded with Linux in Germany. Trolltech and Lisa Systems collaborated on an Linux iPAQ solution demonstrated at CeBIT that includes the iPAQ, embedded Linux, Trolltech's Qt/Embedded and Qt Palmtop Environment (and PIM suite), custom software components from Lisa Systems for power management, internet access, MPEG player, and a KDE-based browser and file manager. The package also includes support for wireless web access using either a Nokia cardphone or a GPRS cell phone.

Esfia demonstrated a Linux-based Chinese PDA solution for the iPAQ at CeBIT that includes Esfia's RedBlue embedded Linux operating system and RedBlue/Palmlike windows launcher, an FLTK GUI application framework, the ViewML browser, PIM applications, power management, internet access, MP3/MPEG player, picture viewer, fax, e-mail, Chinese handwriting recognition, synchronization from IrDA and USB, games and a file manager.

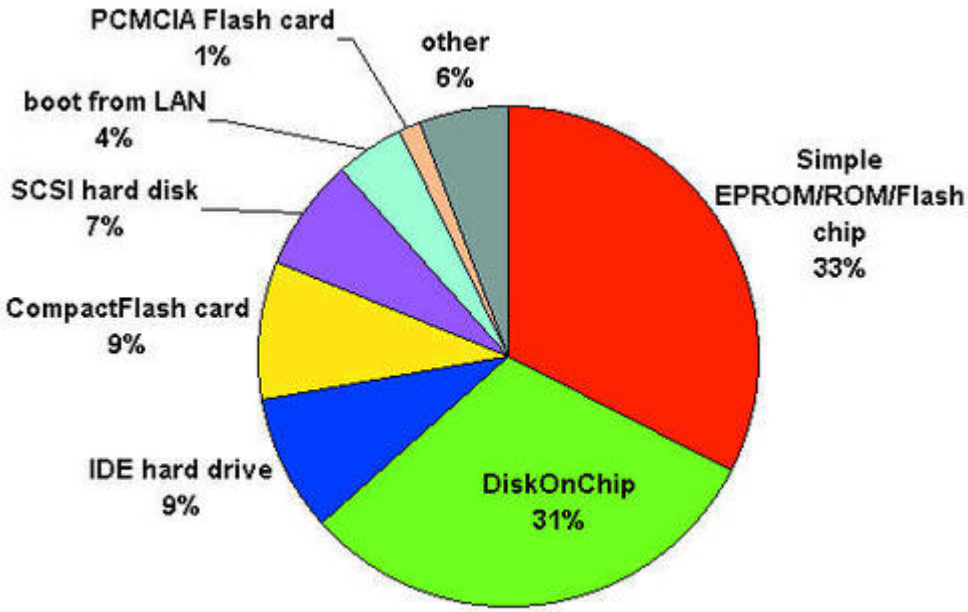
It's in the Numbers

Here are three interesting results from LinuxDevices.com's 2000 survey of embedded Linux developers.



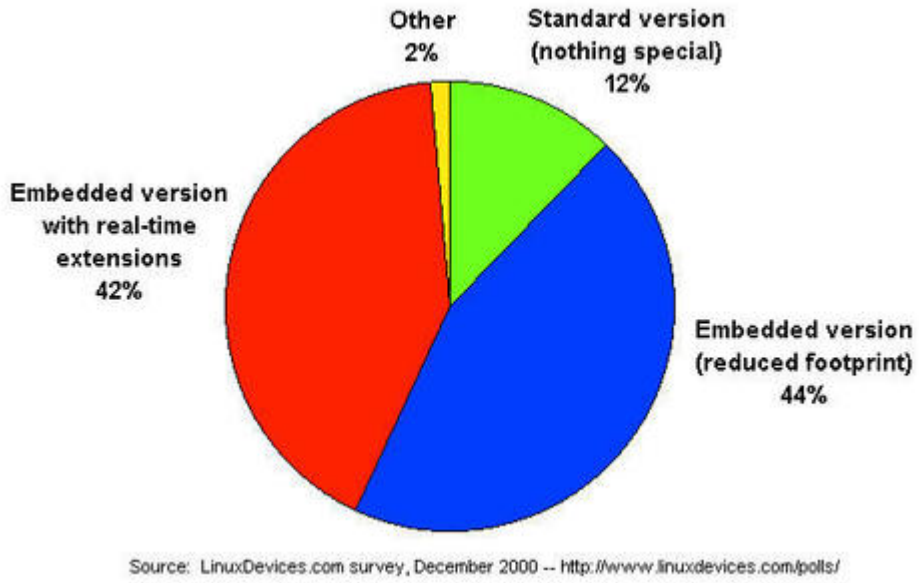
Source: LinuxDevices.com survey, December 2000 -- http://www.linuxdevices.com/polls/

What type of CPU will you use in your embedded Linux-based project?



Source: LinuxDevices.com survey, December 2000 -- http://www.linuxdevices.com/polls/

From what source will your embedded Linux-based device boot its operating software?



What type of Linux OS will your project require?

Please vote in the new 2001 Embedded Linux Market Survey, even if you participated in last year's survey: www.linuxdevices.com/polls.

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com “embedded Linux portal”, which is now part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in launching the Embedded Linux Consortium.

Archive Index Issue Table of Contents

Advanced search

Journalism 2.0

Doc Searls

Issue #86, June 2001

“Journalism is the one solitary respectable profession which honors theft (when committed in the pecuniary interest of a journal) and admires the thief....However, these same journals combat despicable crimes quite valiantly when committed in other quarters.” —Mark Twain

Linux for Suits

Journalism 2.0

by Doc Searls

Journalism is the one solitary respectable profession which honors theft (when committed in the pecuniary interest of a journal) and admires the thief....However, these same journals combat despicable crimes quite valiantly when committed in other quarters. —Mark Twain

On page 33 of the March 5, 2001 issue of *eWeek* magazine is a full-page story by Grant DuBois and Roberta Holland with a rather wordy title:

> Emerging Tech // Peer to PeerLooking before they leapIT taking a wait-and-see stance on Jxta and .Net.

At the center of the page, just over the headline block, a large photograph bears the caption, “Brian Moura in San Carlos, California, questions the usefulness of P2P.” Here's how the piece opens:

> Brian Moura needs to hear better arguments in favor of peer-to-peer networking technology before he gives it any consideration.
>
> “Peer-to-peer is not on our radar screen”, said Moura, assistant city manager in San Carlos, Calif. “If we already have application servers and file servers,

> what's the advantage or value-add of letting each person's PC be a server?"

The whole piece is impelled by Moura's skepticism, which would be fine if Moura legitimately exemplified all of IT—or at least a significant wedge of the profession—but he doesn't.

Type "Brian Moura" into Google and hit the "I'm Feeling Lucky" button, and you'll go straight to Moura's home page. The top link there takes you to the San Carlos home page, for which Moura is webmaster. The San Carlos site (http://www.ci.san-carlos.ca.us/) has barely changed since Magellan (long since absorbed into Excite) gave it three stars in 1996. The site brags about this distinction, along with its "Best of the Web" award, also issued, fossil records show, near the dawn of Internet time. To be fair, the site itself is useful, as municipal sites go. The update page shows that somebody's trying to keep it fresh.

It's pretty clear from both the site and the quote above that Moura's main purpose in the piece is to substantiate a negative spin about P2P, an acronym the writers peg on Napster, Microsoft's .NET and Sun's Jxta. Ironically, the most widespread P2P phenomenon—the one where most of the action is taking place—is the web log movement. At their most institutional, "blogs" include downtown hacker hangouts like Slashdot, Kuro5hin and Advogato. At their most individualistic, they include dozens of thousands of public journals, most of which are hosted by Blogger, Userland or Pitas.

As Glenn Fleischmann put it in the *Seattle Times* recently, "Some blogs are closer to public diaries; others, the idiosyncratic or authoritative musings of experts and cranks." Together they constitute a vast peer-to-peer network that performs a constant story-finding and fact-correcting mission for both their own readers and mainstream journalists who don't want to be caught fluffing nothing into something.

Thanks in large measure to blogs, the journalism of the future will be fed mostly by a peer-to-peerage of linked and syndicated writers and sources who endlessly report facts, stories and informed speculation. The nature of peer-review journalism is very much like the nature of peer-review software and, naturally, includes many of the same people. News, ideas, facts, jokes, interesting links and other items are constantly vetted, checked, challenged, credited, linked and spread. Underlying all of it, the main purpose is no less social than open-source code development: to share what we know and what we think. The goal is not to attract interest with bogus stories or to attract readers to advertisers—two common hidden agendas of traditional journalism, especially trade publications.

More and more journalists have blogs and use them as instruments of their professional work. This is the case with Glenn (who writes for many publications), with Dan Gillmor of the *San Jose Mercury News*, with Deborah Branscum of *Newsweek* and *Fortune* and with a certain senior editor for *Linux Journal*. In some cases, we even vet nascent stories or hunches that might become stories, inviting responses that help inform those stories. That's what I did recently when Eric Schmidt ceased presiding over Novell's ongoing failure and became “part-time” president and CEO of Google. Glenn's account of what followed became part of his *Seattle Times* piece:

> One experience I had recently illustrates how a blog works. Doc Searls recently posted musings (doc.weblogs.com/2001/03/27) about whether Google's (http://www.google.com/) hiring Eric Schmidt as chairman might result in the purchase of the search engine by Sun Microsystems, a company Schmidt helped found. I sent Searls some pithy comments and a number of others did the same. Searls posted an update the next day with our responses (doc.weblogs.com/2001/03/28). He also linked to an on-line-only column by Branscum at *Fortune* magazine's site about Google's success working with on-line advertisers.
>
> This kind of mild dust-up happens all the time, with a mix of journalists, ordinary readers and subject experts responding to their colleagues with no intermediation and little compunction. These responses are incorporated into blogs, resulting in more cross-links and a richer vein of detail.

The responses are also incorporated into the writer's stories and editorials, as Glenn and I both illustrate. Today I understand a lot more about Google, thanks to responsive blog readers that include a number of company insiders who are not my usual PR sources. So when I do write something, I'll have a lot more sources to call upon and information to work with.

Note that this doesn't change the nature of mainstream journalism one bit, except to make it more resourceful. The growing difference here, as with the Open Source movement that both surrounds and populates the software industry, is in the involvement of many more people in a web of trust and respect. It also resembles, in a raucous and noisy way, the traditional marketplace we call a bazaar.

I was delivered this realization recently when I sat in a plane next to an amazing gentleman from Nigeria named Sayo Ajiboye. A deeply thoughtful religious scholar who took eight years to translate the highly annotated *Thompson Bible* into his native language of Yoruba, he compared what I told him about both Linux and blogs to traditional marketplaces in his country. “Markets are not just

about business", he said. "They are about relationships." In fact, he went on to say, they are constituted by relationships, motivated at least as much by the desire for relationships as by the desire to sell and buy.

Old media never go away, they just obtain larger contexts. Mainstream journalism will always involve publishing. But surrounding that mainstream will be a watershed of other people—journalists in the literal sense and their readers. The whole watershed becomes a marketplace; not just for shared passions, but for trust and authority.

A case in point. In February, Dan Gillmor made the mistake of quoting something Richard Stallman was drafting and sharing with a few other folks by e-mail, including Dan. When Richard and other peers caught the mistake, they called him on it. It wasn't a big deal, but Dan quickly admitted it, removed the web copy of the offending piece, and everybody moved on. In the course of the matter, Dan's authority increased.

The growth of highly cross-sourced journalism by folks like Glenn and Dan is moving toward the norm for journalism both in print and on the Web. It makes traditional newswire-fed journalism seem increasingly anachronistic, as well as unreliable. CNET, for example, is a source of many good stories but some provocative clunkers too. That's what we had in February when CNET ran a poorly sourced and credited *Bloomberg News* story in which Microsoft's Jim Allchin said some unkind things about open-source development and licensing. The story stirred up a huge fuss, but it lacked both context and authority in the literal sense. Who exactly spoke to Allchin? Where? About what? None of this was clear.

It's not just a coincidence that CNET leads the way in deploying huge new reader-averse, 240 x 400 "interactive marketing units" (yet another buzz phrase for advertising). Nor is it a coincidence that a current *eWeek* news piece about these huge new ads fails to include a single source from the opposition. The headline in the magazine reads "Online Advertising: Bigger Is Better". But the headline on the Web in ZDnet reads "Online Advertising: Will Bigger Be Better?" Interesting difference, no? The last words in the piece are given to Beth Eason, VP and GM of DoubleClick. "Somebody needs to pay for the Internet", she says. "Advertising needs to be the economic engine that drives it."

The Internet she's talking about is what remains of a massive investment project proving to be little more than a fantasy. The Internet that existed before that project showed up persists uncorrupted. Its current conversational manifestation is P2P, which has nothing to do with advertising and is not driven by Microsoft, Sun or other large potential advertisers.

This kind of journalism will die of exposure, right along with the advertising projects its publishers continue to fund under the illusion that its constituency is an audience, rather than a bazaar filled with sources and fact checkers.

Also due for a change are stories about conflict for its own sake. Of course, conflict is what makes stories interesting. Without it you don't have a story. After all, stories never start with “Happily ever after” (if they did they'd be press releases). Conflict is what journalists naturally like to find and cover. Too often, though, it gets invented. That's what Stephen Shankland did recently in a CNET story about Maxtor dropping open-source products in favor of Windows. It's also what DuBois and Holland did in their P2P piece in *eWeek* and ZDnet.

The purpose of both stories was less to report than to conjure. The CNET piece conjured the illusion that there's a big fight between open source and Windows in the embedded applications market. The *eWeek*/ZDnet piece conjured the illusion that IT managers have some kind of problem with P2P.

The author Peter Rengel (who writes about sex and intimate relationships) once told me, “we choose the levels of truth at which we are willing to live,” and “the challenge is always to go to the deeper level.” He added, “there's always a deeper level.”

Journalism-as-usual lived at a certain level in a world without the Web. In that world, there wasn't a peerage of highly open sources. To live in the new world, it has to work at a deeper level, or it won't survive.

Doc Searls is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*.

Archive Index Issue Table of Contents

Advanced search

Heroes of Might and Magic III for Linux

J. Neil Doane

Issue #86, June 2001

A game whose addictive properties can, quite literally, suck you in until your mouse-hand is spasming involuntarily and your eyeballs are glazed and crusty.

- Manufacturer: Loki Entertainment Software
- E-Mail: sales@lokigames.com
- URL: http://www.lokigames.com/
- Price: $29.95 US
- Reviewer: J. Neil Doane

It used to be that turn-based games were all the rage, but somehow it seems that in a world filled with first-person shooters and real-time strategy games, they've fallen through the cracks of general popularity. Loki's Linux port of 3DO's *Heroes of Might and Magic III: The Restoration of Erathia* sets about to fix that little oversight with a vengeance in a game whose addictive properties can, quite literally, suck you in until your mouse-hand is spasming involuntarily and your eyeballs are glazed and crusty. I mean it, if you believe you might have an addictive personality, if you have someone in your household who might actually want to speak with you in the next few days, if you don't believe your boss will accept, "...can't talk now...game..." as a valid sick-day excuse, think twice before sitting down to a *Heroes III* session. Fact is, it's the turn-based nature of the game, the part of *Heroes III* that seems so benign on the surface compared to its faster-paced cousins, that erodes your conscious mind and turns it into a soft, pliable sludge. You begin thinking in terms of how you only need a few more turns to set yourself up for a another small battlefield victory or strategic advantage that leads to a new perspective, a new strategy, which leads you to believe that only a few more turns are needed to set yourself up for something else; and so the vicious circle sucks you in as you hurl yourself onward toward conquest (or sleep deprivation-induced insanity). *Heroes III* is a game of strategy, chance, tactics and luck; a mixture of some of the best elements from *Dungeons and Dragons*, *Chess*, *SimCity* and *Warcraft* blended

seamlessly into an RPG that makes you the leader of armies of mortals and supernatural creatures fighting against warlocks, wizards, warriors and witches (and worse).

The Story

Heroes of Might and Magic III is the latest installment in the epic *Might and Magic* universe of games and takes up where *Heroes II*, the ninth title in the saga, leaves off. This time you are on the continent of Erathia where a complicated tale of murder and conquest unfolds as Catherine Ironfist seeks to investigate the death of her father and restore Erathia to its former glory. The story naturally unfolds through movies and cut-scenes with narratives that explain each new situation in a compelling and easy-to-follow format.

Basically, the idea is you control a cadre of “heroes”, the game's main characters, as well as the settlements they conquer, the resources they appropriate, the lands they explore, the armies they command and the treasures they find. Your heroes' goals are typically stereotypical *D&D*-style objectives (“kill the golden dragon queen” or “find the grail before your enemies do”), and as they progress and mature, they gain experience, new powers, skills, knowledge and new warriors to take into battle. The scope of the game is nothing short of Homeric at times; although you can choose to engage in shorter single-player games if you wish, often even medium-sized quests will last well into the small hours if you aren't careful and in campaign mode. Your characters can grow and mature through a long series of goal-oriented games to reach a final objective that may take many days and nights to complete.

Figure 1. Exploring the Kingdom

Game Play

As stated before, *Heroes III* is turn-based; more specifically, each turn lasts for one Erathian day. One day in which you are allowed to control each of your heroes in any way they can be controlled (fighting, moving, teaching, learning, exploring, acquiring resources, trading resources, etc.); and manipulate the activities in the cities under your control (build new buildings, recruit new warriors or heroes, buy or sell merchandise or artifacts, etc.). When you're done with your day's activities, you complete your turn, wait for your opponents to complete their daily events (computer opponents take only moments to make their moves) and then begin the next day's events. Warrior resources in each city are replenished once per week, so if you go out and lose most of your troops in a battle on Monday, you'd better hope you can hold off your attackers for a week with what you have left.

If you think the game sounds complicated, you're right, it is. The learning curve is pretty steep and an intimate look at every detail of the game requires a rather serious study of the included 133-page manual. However, a 12-page PDF tutorial guide is included on the CD, and the built-in tutorial mission takes only about 45 minutes to complete. Afterward, one should have a very solid command of the fundamental game options available and a firm enough handle on the game's mechanics to jump into play and learn the specifics. In fact, for all its complexity, the game interface is intuitive enough that many players will likely find they can head right into the tutorial game and learn enough in a few minutes of poking around at the assorted controls to get themselves on the right track and playing single-player games with relative ease. What's more, the developers took into account the complexity of the game and scattered contextual help liberally. Right-clicking on virtually any place in the game will produce a pop-up window describing in succinct detail the area, button or icon currently under the pointer.

Game play focuses exclusively on the actions of the heroes themselves and the cities under your control. Heroes are responsible for exploring the areas around the cities, locating any of the nearly 130 different artifacts available, finding resources like mines or other sources of income for the city, as well as battling to defend the cities from attack, conquering new cities and fighting other heroes to expand your holdings. Cities produce gold currency from resources you provide them, which you then use to buy buildings and recruit heroes and warriors. Warriors provide valuable services to your heroes and act as a clearing house for all manner of soldiers or creatures your heroes may need in order to beef up their armies.

Figure 2. An Early Fortress Town

The cities themselves come in eight different varieties and are made up of dozens of individual structures, each customized for the type of cities they are built in. For example, in the “Castle” city-type, one will find buildings such as a guardhouse (where pikemen can be recruited), a monastery, training grounds (the favorite hangout of cavaliers) and even supernatural structures like a holy “Portal of Glory”, where one can recruit powerful Archangels for the right price. The flip side would perhaps be the “Inferno” city-type, where such sinister structures as fire lakes can be built to supply supernatural Efreet, kennels built to supply Hell Hounds and the deadly Cerberus, and even a “Forsaken Palace” from which Archdevils are called into the service of your army. Other cities are homes to wizards and genies, Pegasus and dwarves, even dragons and ogres. In total, there are 118 different varieties of creatures at your disposal, and each has specific attack characteristics and features designed to facilitate various tactical roles on the battlefield, from ranged attacks at a distance to close-quarters hand-to-hand combat or anything in between. Each building has a specific cost associated with its construction and some characters and buildings can be used to supply materials or resources. For example, blacksmiths can forge weapons; libraries can enhance the knowledge of visiting heroes; mage towers can be used to teach visiting heroes new spells and incantations; and new heroes can always be found loitering in the local taverns, waiting to be hired to command armies for your kingdom.

There are 128 different individual heroes, each with his or her own unique traits, strengths and weaknesses. These heroes lead your military might, exploring the lands with the armies you've placed under their control, claiming resources for your kingdom and protecting your sovereign territories from hostile invaders. Heroes themselves do not participate in battles, but each has various types of spells, knowledge and skills they can use to drastically help the

troops under their command. When in battle, a turn-based combat begins and a close-up view of the hexagonal-grid battlefield is presented, as shown in Figure 3.

Figure 3. Combat Screen

Each side takes turns in combat until a victor is determined, by either one hero vanquishing another's armies through total destruction or by one side surrendering or retreating. In the case of a surrender, the hero runs away in shame, and you'll have to recruit him or her again to get them back into the service of your army. In the case of a defeat or a retreat, the shame is simply too great, and the hero will abandon your cause altogether. Victors of combat receive experience points that will help them gain new levels of maturity and learn new knowledge and skills that can help them in battle or in exploration: skill in tactics can give you free moves at the beginning of combat, skills in navigation can help you move farther each turn while in a ship, skills in diplomacy will help your heroes convince others they meet to join their armies and so on. Combat against fortified cities takes place in a special siege-combat screen, where an attacking hero's army has to bombard and destroy a city's ramparts with catapults in order to physically invade.

The graphics in *Heroes III* are superbly detailed, colorful and complex. The various interfaces are intuitive and ergonomic, which is fortunate since players will be sitting spellbound and using them for significant fractions of their waking moments. The sound effects and background music are where *Heroes III* really excels; the music actually changes to fit the mood of the game and does it often, at all the right moments. Exploring your kingdoms, you'll hear classical symphonies and powerful soaring adventure music; go into battle and the score changes to something more foreboding and suspenseful. Head into a castle city and the music is an Arthurian harpsichord concerto; head into a

dungeon and the music is an eerie melody with deep oboes and cathedral bells —you get the idea. The music adds much to the ambience of the game and never seems to get old or repetitive.

Multiplayer modes are, historically, one of the best parts of turn-based games and *Heroes III*'s multiplayer action is, unfortunately, still back somewhere in history. While TCP/IP modes of play are available, no functionality exists for connecting to any type of centralized, on-line community to manage multiplayer games. You will need to find the IP address for your *Heroes III* game host with some other method, and then connect directly to it. Alternately, you can play "hot seat" games, wherein one person sits in the hot seat and makes their moves, gets up and lets the next person play. There is no option to play by e-mail, like a lot of turn-based titles, so to play in that fashion requires players to exchange hot seat files (saved hot seat games) with each other in order to remotely do turn-based, long-term gaming. In this reviewer's opinion, this is really one of the only bad points about *Heroes III*, and an area that was sorely overlooked by 3DO.

Linux Notes

Tested on a Matrox G400-based, 500MHz PIII system with 128MB RAM, this game performed flawlessly. I suspect that it would function quite well on a much lesser system as there is no requirement for 3-D acceleration, and the calculations for combat and movement really shouldn't be that complicated. Loki's recommended minimums for the game are 32MB RAM and a Pentium 133. You'll need an OSS-compatible sound card, as well as an X server capable of doing 800 x 600 in 16-bit color. The install footprint is highly customizable since the installer allows you to separate out all the major components and leave in any of the scenarios, sounds, graphics, music or videos on the CD if you like. This allows *Heroes III* to take up as little as 5MB to as much as 350MB of your drive's real estate. Depending on how much data you leave on the CD, you may need more than the recommended minimum 4X CD-ROM drive; running the videos from such a slow drive might cause some rather nasty stuttering/ rebuffering. As usual, Loki recommends at least any of the 2.2 kernels or above for their game ports.

There are some small problems with the default install. For instance, the version of *Heroes III* that Loki ships doesn't do full-screen mode by default (or at all as any user other than root from what I could tell). However, Loki's single upgrade patch is only about 1MB and fixes many small bugs, including the full-screen mode problem, and is highly recommended to be applied over the default install before playing. Interestingly, this patch also includes hooks to AAlib, the ASCII Art Library, which supposedly allows truly desperate *Heroes III* players without X to get their fix from the console in glorious ASCII detail.

Additionally, Loki is also offering a beta version of their *Heroes III* map editor for Linux, which allows you to create your own *Heroes III* adventures. They are also offering a free 17MB demo of the game from their web site, http://www.lokigames.com/.

Summary

Plain and simple, this game is exquisite: turn-based gaming at its finest. The interface makes sense, even if you've never played an RPG-type game before, and the quality of the workmanship, attention to detail and overall polished feel of the game makes everything seem to work the first time, the right way. Although the game is rather complex, it's straightforward and more importantly, fun to learn. *Heroes III* is one of those games that you sit down to tinker with and end up leaving your desk eight hours later because you have to go to bed sometime. Highly recommended for penguins of all ages.

J. Neil Doane (caine@valinux.com) is a professional services engineer with VA Linux Systems and an Indiana escapee. Between prolonged spasms of rabid geekness, random hardware scavenging and video gaming, he is a pilot, a guitarist and a very poor snowboarder.

Archive Index Issue Table of Contents

Advanced search

PogoLinux RAID Workstation

Choong Ng

Issue #86, June 2001

For this review, PogoLinux sent me one of their new RAID workstations, the PogoLinux Velocity.

- Manufacturer: PogoLinux
- E-mail: sales@pogolinux.com
- URL: http://www.pogolinux.com/
- Price: $1,499 US
- Reviewer: Choong Ng

For this review, PogoLinux (pogolinux.com) sent me one of their new RAID workstations, the PogoLinux Velocity. The system came configured with a 1GHz Athlon processor, 256MB of PC133 SDRAM and two IBM 45GB hard drives attached to a Promise ATA100 RAID card in a RAID 0 configuration.

For those that are unfamiliar with RAID, RAID stands for redundant array of inexpensive disks. The idea is that one may provide enhanced reliability or performance by using an array of relatively inexpensive disk drives combined with a special hardware or software controller. The three most common RAID configurations are called RAID 0, RAID 1, and RAID 5. RAID 0, also known as striping, is the division of data between drives such that a single read or write operation may combine data from two or more hard drives, thus increasing the

aggregate transfer rate. RAID 1, also known as mirroring, does the opposite of RAID 0: instead of focusing on performance, RAID 1 focuses on increasing reliability by storing multiple copies of your data on different drives, thus reducing the chance of a drive failure destroying your data. I opted to have the workstation preconfigured with RAID 0 for the enhanced performance.

Setup and Initial Impressions

Setting up the Velocity was a breeze: five minutes to get it out of the box and onto my desk and another ten to boot it up and configure it to talk to the network. Those of you who will want to add custom hardware such as a preferred graphics board, gigabit Ethernet or a DVD drive will appreciate the Velocity's maintenance-friendly case with just two thumbscrews between you and the motherboard. Red Hat 7 will recognize most third-party hardware without problems.

There are two big issues to be aware of before you set up the system. One is that the default Red Hat 7 has a long list of known problems, and because of this you may have trouble getting some applications to run properly. The other major issue that I ran into is that the kernel needs special parameters to boot properly from the Promise RAID hardware. This only becomes an issue if you want to replace the kernel or install an alternative Linux distribution; Pogo's default configuration works fine. The usual symptom is a kernel panic when the kernel attempts to mount partitions, but the fix is simply to provide the kernel with the correct I/O addresses. For a detailed explanation of how to do this see Aaron Cline's “Unofficial Asus A7V and Linux ATA100 Quasi-Mini-Howto” at http://www.geocities.com/ender7007/. Once the machine recognizes the card no further configuration is necessary to take advantage of the RAID controller, and other than the issues just mentioned, I had no problems with setup.

My overall first impression is that the Velocity is a very speedy machine—as one would expect from any 1GHz Athlon box—and fairly well put together. As someone who frequently swaps hardware out of my machines I also appreciate the ease of opening the case via the thumbscrew-attached side panels. Having set up the machine and played with it for a little bit, it is time to measure how fast the Velocity really is.

The Benchmarks

For these benchmarks, the PogoLinux RAID box in RAID 0 mode will be compared to a similarly configured non-RAID system, same motherboard and processor (well, A7Pro vs. A7V), same RAM and a similar ATA 100 drive (but just one for the non-RAID system).

Listing 1. PogoLinux Velocity (RAID 0)

Listing 2. Single Disk System

Using the Bonnie benchmark suite, the PogoLinux box produced some interesting results. Tests that performed large numbers of very small disk transactions did very poorly on the PogoLinux RAID box—about half as fast as the comparison system—while tests that rely on fewer transactions involving larger amounts of data did much better, right in the middle of the 40-50MB/s transfer rate quoted by PogoLinux. This information is very important when one is considering the purchase of approximately $1,578 worth of hardware. So, on to a real-world test.

To compile Mozilla, use the time command for measurement, as shown in Table 1.

Table 1. Times (mm:ss.s) and Improvement (%)

In most applications, the RAID system didn't perform noticeably differently from the single drive system. Some tasks, such as untarring Mozilla, actually came out slower on the RAID system. Compiling Mozilla was only faster by about ten seconds out of 50 minutes. This is indicative of a near-universal truth in computing: aggregation can buy you increased bandwidth, but it can't buy decreased latency. The likely explanation for why the RAID 0 system has relatively poor performance when performing many small accesses is rather lengthy, but suffice it to say that it is related to the fact that having to wait for two read/write heads instead of one increases the array's average seek time.

What this means in the real world is that having a RAID 0 disk system will only accelerate tasks where large amounts of contiguous data is transferred (where increased transfer rates help) and not tasks that require many small disk accesses (where disk latency is more important than transfer rate). One good example of this is working with applications that use large data files, such as editing high-resolution photos in the GIMP, where I did indeed notice significant improvement in loading and saving large files.

Other tasks that benefit from increased disk transfer rates include database searches, video editing or any type of high-bandwidth data capture (i.e., direct-to-disk audio and video). Tasks that won't benefit from RAID 0 (especially as opposed to having two drives operating independently) include file servers serving many concurrent users making small requests, database servers supporting similar workloads, most general-purpose applications and development software, etc.

The Bottom Line

For many users' purposes, a RAID 0 system will provide less performance benefit than spending the equivalent money on getting a single fast hard drive, faster processor, more RAM or a faster motherboard. The price difference between PogoLinux's Velocity workstation and a similarly configured workstation without RAID, such as PogoLinux's Altura workstation, is about $300 if you still buy another 45GB IBM hard drive. For $300 you can buy a 1.2GHz Athlon (instead of 1.0), an extra 256MB of RAM (or an extra 512MB if you stretch it to $370) or a 64MB GeForce2 GTS (instead of the included 32MB MX). On the other hand, if you really need the maximum throughput for working with large files or the reliability of a RAID 1 configuration, the RAID system is definitely the way to go.

The Good/The Bad

Choong Ng is the product reviewer at *Linux Journal* and a teriyaki connoisseur.

Archive Index Issue Table of Contents

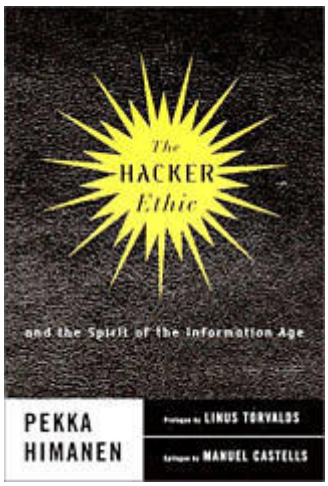
Advanced search

The Hacker Ethic

Michael Stafford

Issue #86, June 2001

Even if you consider yourself a well-versed hacker, you'll enjoy reading about your spiritual roots as well as the positive social implications of the hacker ethic.



- Author: Pekka Himanen with Linus Torvalds and Manuel Castells
- Publisher: Random House
- URL: http://www.randomhouse.com/
- Price: $24.95 US
- ISBN: 0-375-50566-0
- Reviewer: Michael Stafford

The Hacker Ethic is a far more significant work than its size lets on. It attempts to do no less than explain to the general reader what it's like to think, work, create and play like a hacker. With so much confusion and bad press around hackers, this is no small challenge. Pekka Himanen situates the hacker worldview historically, contrasts it with the modern, work-centered worldview and speculates on the contribution to humanity the hacker ethic can make. Even if you consider yourself a well-versed hacker, you'll enjoy reading about

your spiritual roots as well as the positive social implications of the hacker ethic. I might just give copies to my family and friends so they better understand why I live, work, sleep and believe the way I do.

This book stems out of a collaborative friendship between Pekka Himanen, Linus Torvalds and Manuel Castells (philosopher, hacker and sociologist, respectively). It's only fitting that a book on the collaborative hacker ethic should in fact be jointly written. Torvalds contributes the prologue, presenting in a nutshell what makes hackers tick. Himanen writes the largest, middle section. And finally, Castells' epilogue rounds out the book with a discussion on the information age's social implications. Well researched endnotes and bibliography are included.

To explain why hackers would work together without pay to make something like Linux, Torvalds pens “Linus's Law”. With tongue somewhat in cheek, Linus's Law posits progress and evolution as upward motion along a hierarchy of motivations: from survival, to social life, to entertainment. What makes hackers tick, according to Torvalds, is this last and most sublime motivation. By entertainment, he's not thinking of games so much as “the mental gymnastics involved in trying to explain the universe”, whether you're Einstein, an artist or a hacker. Coding an operating system is, after all, a great deal like explaining the universe because one codifies the parameters and elemental models of everything that can happen. Torvalds means entertainment to encompass the passion and playfulness of activities intrinsically interesting and challenging—stuff we stay up late working on because we love doing it, not because of a deadline.

Wasting no time, Himanen makes the distinction in his first paragraph between hackers and crackers. Readers are to understand that it is only the latter who use computers to commit crimes, bring down e-commerce, produce viruses and generally sow mayhem. By “hacker”, Himanen means those who “program enthusiastically” and who believe, as Eric Raymond describes in *The Jargon File* (http://www.catb.org/jargon/), that “information sharing is a powerful positive good, and that it is an ethical duty of hackers to share their expertise by writing free software and facilitating access to information and to computing resources whenever possible.” Maybe the lines can't be so easily drawn. I'm prone to think that cracking a system's security can sometimes be an ethical, populist act. Himanen does concede that “most computer hackers support only some parts” of the general ethic he's describing. Still, it is fair to assume with Himanen that a general, cohesive hacker ethic and its effects can be clearly discussed. It is to Himanen's credit that he can meaningfully characterize a group largely defined by the primacy it grants to independent thought.

How does Himanen bring readers into the hacker's world? He starts by looking at hackers' relationship to a cultural universal: work. To describe the hacker work ethic, Himanen uses as a counterpoint a landmark treatise on modern work and culture. What Max Weber described in *The Protestant Ethic and the Spirit of Capitalism* is so much a part of our way of experiencing work now that it seems as if things must have always been so. Calling attention to our modern work ethic feels almost like pointing at the air around us, until you realize that this pervasive work ethic has all but suffocated us. In contrast, the hacker ethic is a deep breath of fresh air. As Himanen refers to Weber, he shows us that before the Reformation, attitudes about work and its cultural importance were quite different and came to change with the spread of the Protestant business model. The hacker work ethic has potentially the same possibility to culturally affect the experience of work. At the very least, the hacker work ethic insists on there being an alternative mode of experiencing work. Himanen makes the case that in this information economy, the hacker's mode of work is also pragmatically superior to the protestant work ethic: “The information economy's most important source of productivity is creativity, and it is not possible to create interesting things in a constant hurry or in a regulated manner from nine to five.” Workers can't freely and creatively pursue the projects of their passions if they are busy jumping through hoops and punching a clock several times a day.

As the hacker ethic extends into the social and political arenas, Himanen isolates expression and privacy as the core hacker ideals in the public sphere, particularly on the Net. They hold importance because they are vital in the resistance to exterior regimentation and over-management. In the public sphere those forces come from government and business. Government regulation and control of expression, however well-intentioned at the outset, will always contribute to the message that individuals are not mature enough to use communication without guidance and supervision. The worker who cannot work without supervision also cannot participate unsupervised in the public sphere. With privacy unguarded, private businesses can and do construct intimate and complete consumer profiles of individuals that are then for sale to the highest bidder. In this way it's possible to scrutinize the lifestyles of workers and job applicants. The message here is either you're not smart enough to be a good consumer on your own without our guidance, or employers not only can regiment your work time, but also have a say in how you spend your time away from work. Himanen recounts stories of renowned hackers fighting for expression and privacy in both small, peaceful ways and very dramatic ways, such as in Kosovo where they played a decisive role in keeping free expression alive.

The Epilogue by Manuel Castells ends the book with a densely theoretical discussion of the societal forms he sees emerging from the new economic

infrastructure brought about by recent technology—a novel application of the Marxist idea of infrastructure determining cultural forms. Castells' portion of the book was for me the most demanding and intellectually rewarding. He convincingly shows how companies, industries and even nations are functioning and structuring themselves like dynamic networks, where workers and managers are constantly reprogramming themselves to perform new tasks and new goals. In these networks, the real employers of workers aren't companies so much as projects within a network. As these networks become increasingly efficient and object-oriented, a worker's relevance and permanency are guaranteed only by his or her ability to self-reprogram. For most workers, technology has mainly served to optimize the amount of useful work that can be squeezed out of a day. Whereas Ben Franklin said “Time is money”, now with the compression of time, Himanen writes that “even shorter units of time are money”.

The Hacker Ethic offers not only an able description of the hacker worldview, but also points to various directions in which hackers can work to help counteract some of the inhumane consequences of our network society. These directions include integrating outsiders (back) into society's network through widespread net access and education, initiating circles of hacker creation where useful tools result from a mutual learning process, aerating the experience of time for oneself and others to allow more space for creativity and championing the causes of expression and privacy.

Michael Stafford received his MA in French Literature from Duke University. At UCLA he taught French and developed educational web media. Feeling a career in the academy would be too risky, he became a commodities trader and trading systems designer. He has won various awards as a trader and openly shares many of his analysis tools. He can be contacted at michaelstafford.com/LJ.htm.

Archive Index Issue Table of Contents

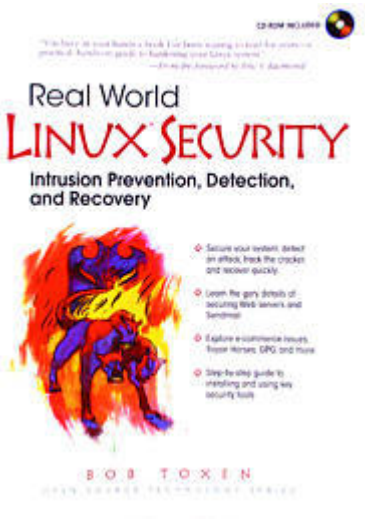
Advanced search

Real World Linux Security

Don Marti

Issue #86, June 2001

This book is here to help you, not scare you, and you should be able to get through the most important parts in a weekend.



- Author: Bob Toxen
- Publisher: Prentice-Hall
- URL: http://www.realworldlinuxsecurity.com/
- Price: $44.99 US
- ISBN: 0-13-028187-5
- Reviewer: Don Marti

Real World Linux Security is the kind of book to which we have to give a good review, as it is seemingly written to butter us up. Bob Toxen says most Linux distributions install too many extra dæmons by default, he lists privacy-violating web advertiser DoubleClick, Inc. as a security issue, and he even uses http://www.linuxjournal.com/ as one of the hosts in an example. We like him already.

We also have to like the concept of a big, fun workbook full of things we can do to increase the security of our Linux systems and how to prepare to get back up with minimum pain if they do get compromised. So please resist the temptation to, after taking one look at these 694 pages of cracks, sploits, bugs and vulnerabilities, go home, unplug your Linux box from the Net and crouch behind it with a shotgun. This book is here to help you, not scare you, and you should be able to get through the most important parts in a weekend. There's no cause for alarm, but no reason to be smug either.

Do you really need this book? We'd like to be able to say you only need it if you offer network services or develop software. In the future, when Linux distributions start to be a little more sensible about their default security policies, you might be able to get by without this book, if you install a locked-down, client-only configuration and subscribe to a mailing list for occasional security updates and bug fixes. But in the future you'll have an apartment on the Moon, too. This is still the wild and wooly Linux scene of Earth, 2001, and every checklist network feature printed on the side of the shrink-wrapped distribution package means more work for you—something you'll have to secure or remove.

Before we get to the good parts of *RWLS*, let's start with the savage criticism, shall we? First of all, the book mentions browser security but isn't up to the current state of the art in privacy tools. It's good for Toxen to mention that DoubleClick is a security risk, but setting up blackhole routes to some of their servers just gives you something to maintain when DoubleClick moves them. It isn't going to be as effective as making your nameserver authoritative for the doubleclick.net domain or, better yet, running a proxy. And there are better ways to manage cookies than just linking .netscape/cookies to /dev/null.

However, this is basically a book for the system administrator, so no big deal. Users can easily find browser privacy hints—the hard part is locking down services. Strangely enough, though, in Chapter 13 Toxen dismisses the entire network time protocol (NTP) with the comment, “Some people prefer NTP, but it uses only UDP, which is too insecure.” Instead of NTP, he recommends using the older **netdate** or **rdate** programs from **cron** and manually calculating the clock drift if a system gets cracked.

NTP has cryptographic authentication capabilities and, even when used without authentication, will only let an attacker who spoofs UDP packets fake out your computer clock v-e-r-y s-l-o-w-l-y. Don't the advantages of having to-the-second synchronized file modification times and log entries on all your systems outweigh the risks of this difficult attack? Why make yourself have to correct the logs when you can least afford the time to do so?

Coming up with a reasonably secure NTP plan, though, is something you should be able to do after you understand the security principles explained in the rest of the book. Hints: use authentication among your in-house NTP servers and those of cooperative ISPs, friends or business associates; use more than one trustworthy public time server; add a radio clock if you're paranoid; and check your logs.

That's about it for the savage criticism. What you can expect to like about this book is a bunch of configuration tweaks, coding tips, preparatory measures to minimize the effects of an intrusion, security philosophy, anecdotes, office politics and tips on how to steal laptops at airports. The book seems to hop around a little, but it's not like you're reading some free-form brain dump here; if you're doing security, you really do have to think about things at more than one level at a time. Toxen's “knucklehead with a laptop and a phone line” who does an end run around the firewall could be anybody—a home user on DSL, or a traveling user who checks into the Embassy Suites in Las Vegas (which I personally recommend if you're ever in Las Vegas) and plugs into the “High Speed Internet Access in Your Room Only $9.95”.

If your organization depends mostly on a firewall for security, the laptop scenario should give you the heebie-jeebies. Toxen does discuss firewalling, but mainly concentrates on host-based security—he points out that many threats come from inside the organization. Early in the book, he introduces the concept of an “attack path” which you, as the system administrator, should design so that an intruder has to break through as many difficult layers as possible on the path to root on an important box. A firewall can be one step on the path, so he does cover IP chains.

Web developers will get a lot of help from this book, both in detail-minded advice about CGI and other web technologies, and in architecture for web sites. Toxen's “one-way credit card server” architecture, where a separate box keeps credit card data outside the main commerce site database, is a cool idea and applicable to other types of sensitive data, too. Defense in depth is the way to go. Just because an evil recruiter can break in and get a list of employee addresses shouldn't also mean she can see their Wasserman test results or SAT scores.

RWLS gets a thumbs-up in the fun reading department, too, especially the stories of how Toxen, as a UC-Berkeley student, got and kept root on their BSD UNIX systems. Organizationally, the sections are divided clearly enough so that you can read the general stuff and the specifics about what you run now, then read sections on other services as you add them or take responsibility for them.

Unless you code and run an ambitious internet site on your own, you probably won't be responsible for all the security tasks covered in this book. It reveals just how broad of a field Linux security really is. You'll certainly find things to go through and check on your own system in order to cut your intrusion risks now, and you'll also find thought-provoking reading for planning future projects with security in mind.

Don Marti is the technical editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #86, June 2001

Readers sound off.

Problems with Paranoia

I'd like to point out a few things in the “Paranoid Penguin” article on DMZ networks (March 2001): RPC is remote procedure call and not RP control protocol; r commands like **rsh**, **rlogin** and **rcp** don't use RPC. In my humble opinion, examples should use runlevel three instead of two because three is the default runlevel for most Linux servers. Changes in level two don't have an effect because Linux runlevels are independents and not additionals as they are in other UNIX flavors.

—Vitaly Karasikvkarasik@ndsisrael.com

Bauer replies: You're right that RPC is remote procedure call and not RP control protocol and that r commands like rsh, rlogin and rcp don't use RPC. But my point was that both RPC and r commands are inappropriate for use on publicly accessible systems, and I stand by that. As far as which runlevel to use for examples, I could have been clearer on that. By the way, Debian defaults to runlevel two.

Korny Question

I hate to bother you with a question for Stew, but since it is a real newbie type of question you can probably answer it without getting him involved.

In his article in the April 2001 issue “Providing E-mail Services for a Small Office” he shares his cron job in Listing 5. I usually like Korn shell and write my little scripts there, but when I only use one > on redirecting output, I only get the last line sent to the file. Is Bourne better about understanding > vs. >> ? According to the way I read his tracking script, he would not get the output in listing 6. I

would have had to put >> on all the lines except the first one if I wanted the output in listing 6. Am I missing something?

—Richard Keatingrkeating@ins.com

So far as I'm aware, > and >> mean the same thing in Korn, Bourne and, for that matter, csh. > replaces the file, >> appends to it. Listing 5 would not produce the output seen in listing 6. Looking at the author's original, it read:

```
#!/bin/sh
ST=/etc/sendmail.st
MS=/usr/sbin/mailstats
MSO=/tmp/mailstats.txt
if [ -s $ST -a -f $MS ]; then
        echo "General Mail Statistics" > $MSO
        echo ""
        echo "local = Mail local to fileserver

        echo "smtp  = Internet mail"
        echo "relay = Mail from/to Sun system

        echo ""
        $MS
        cp /dev/null $ST
fi
echo ""
echo "Mail Filter/Forwarding Statistics
echo ""
/usr/bin/mailstat -l /home/thriftycompany/mail/from
chown thriftycompany /home/thriftycompany/mail/from
cat $MSO | mail -s "Daily Email Summary"
rm $MSO
exit 0
```

Apparently the original was inadvertently modified at some point in the production process. The listing on our FTP site is correct. Our apologies.

—Editor

Searching for Open Edition

Thank you all for the best computer-related magazine in all categories. I eagerly await every issue of *LJ* (we have a subscription at the school where I work as a teacher) and read almost everything from the front cover on. In the April 2001 issue I read a presentation of Kylix, which I have been waiting for, being a Pascal and Delphi programmer, but the price of these two commercial products has made it impossible for my budget. When I read your article “Stop the Presses” new hope came to me as I thought an Open Edition existed for free download. That hope turned to disappointment when I went to Borland's web page and didn't find any mention of an Open Edition, only the two overpriced commercial products. So where is this Open Edition of Kylix?

—Bertil Wergeliuswbertil@tjohoo.se

The free download version of Open Edition is expected to be available sometime in Summer 2001.

—Editor

Thanks

I wanted to let you know that I think you're doing a great job with *Linux Journal*, and the article “The New Vernacular” was one of the most intelligent articles I've ever read in a magazine.

—Jeremy Hayescotk_hayes@hotmail.com

O Captain, My Captain

I enjoyed the review on *Descent 3* from Loki (*Linux Journal*, April 2001). I think I will enjoy this game because I like the thought of controlling a doomed ship and trying to find help from others, as well as investigating sabotage done to the ship and sending out radio transmissions. Everything about this game is quite intriguing. I do believe you sold me!

—Jose CausingSilhouet98@cs.com

Linux Too US-Centric

Paul Taylor (Letters, *LJ* March 2001) complains about SuSE 7.0 defaulting/reverting to a German keyboard. I'm afraid he won't get any sympathy from those who don't live in the US and have to put up with this type of behaviour from the majority of products. The arrogance of US companies means that most systems default to US dates, US keyboards, US spellings and US measurements, often requiring considerable effort by users to “fix” their machines for their own locale. Linux is particularly poor in this area compared with, for example, Microsoft. At least with a Windows OS, if the locale is set correctly on initial install, it stays that way for all Microsoft applications and most other third party applications (although Microsoft's idea of Metric measurements has a strange US flavour—what is “A4 Letter”?). Not so with Linux. I have often been caught out on Linux, particularly with log files, because the dates are formatted wrongly. The Linux crowd do care about the end user, as long as the end user is a techie and an American. I would rewrite Paul's last sentence as: I think the basic problem with most of the software development crowd is that they don't give a rat's arse about the non-US end user.

—Mark Easterbrook, Southampton, UK

Disturbing FX

I read the article “GFX: XFree and Video4Linux” (April 2001) by Mr. Rowe, and I must say, for the first time in my life, an article in *LJ* really disturbed me.

Mr. Rowe presents himself as a person who hasn't followed Linux from the beginning and doesn't know it, but having picked it up he now dislikes it. He says that Xf86Config is “prehistoric”, Xf86Setup is “not good enough” for him, pure text programs are “dated applications”, **dselect** is “primitive” and so on.

“Why our mouse wouldn't work was a mystery”, he says, and he says also that some desktop managers “also include a window manager, but desktop managers have a lot of other features”. It's ridiculous: can one really come to *LJ* and print such things?

It is clear that (after the X server) the window manager is the first component you need to decore your screen, but obviously enough, Mr. Rowe never tried twm, fvwm, olvwm and all the tiny windows managers that can run on the i386 with 2MB of RAM and 60MB of hard disk. Saying that Xf86Config is prehistoric indicated that one must never have tried to tune manually the timing lines on XF86Config, too.

—Franco Favento dei Favento da Triestef.favento@ieee.org

Rowe replies: Franco, sorry my article disturbed you. Thank you for writing to let me know. I like Linux, but I still see things that need improvement. Many involved in the development of Linux and Linux-based software share this opinion. Otherwise, why the ongoing effort? The criticism that bothered you wasn't really about Linux, was it? It was XFree86, a GUI also used by FreeBSD and other operating systems. If we compare the available configuration tools to editing modelines by hand, then you are right that we should appreciate a great improvement. If, on the other hand, we compare to configuration in Windows, Mac and BeOS, then saying XFree86 is primitive is one of the kinder things that may be said. Many users experience distress configuring XFree86. In response to your question, I have used twm and olvm, but have not tried fvwm. Back when my desktop was a Sun Sparc20 I developed multimedia software running on OpenLook (that is, olvm). Open-source software can be responsive to a cry for improvement. And, what better place to point out what we should be thinking about improving in Linux software than *Linux Journal*? My column is not about moving to Windows (as you suggest), but about moving from Windows to Linux. Thanks again for writing. I hope some of my future articles may be more to your liking.

Simple Accents

This is about the keyboard question to get á or ç (Best of Technical Support, April 2001). I think the answers there were too complicated.

Just edit your XF86Config and make sure your keyboard section does not have any “no dead keys” statements. Finally, pick “en_US” instead of “us” which is the default. Now all those *trs intéressantes* combinations will be available using your dead key. It will work everywhere too, even on consoles.

Below is a sample of what that section might look like.

XkbKeycodes “xfree86”XkbTypes “default”XkbCompat “default”XkbSymbols “en_US(pc104)”XkbGeometry “pc”XkbRules “xfree86”XkbModel “pc104”XkbLayout “en_US”

—Hassan Auragiaurag@geocities.com

Motivated

I got cracked on my birthday! Now, I have the April edition of *Linux Journal* on my lap, and I'm about to dig into the Paranoid Penguin article “Battening down the Hatches with Bastille”. Every Paranoid Penguin article is great reading. (The problem in this case is that I only read the article, I didn't actually apply it yet!)

—Paul Aginpagin@kwiknet.net

Archive Index Issue Table of Contents

Advanced search

UpFRONT

Doc Searls

Issue #86, June 2001

Stop the Presses, *LJ* Index and more.

Linux Makes Machine Translation Development Possible

Systran Internet Translation Technologies was born during the Cold War when the US government wanted to translate a large quantity of Russians texts quickly. At the end of the sixties, it became a private company called Systran, located in La Jolla, California.

In the nineties, Systran decided to dump the OS/390 running under MVS and port the whole system to UNIX. By then, PCs had become powerful enough to host the translation engines. An automatic translator was used to migrate most of the assembly code into C code.

The original port was made to Solaris, but they quickly switched to cheaper hardware, PCs and Slackware (they've since moved to Red Hat). The reasons for the choice of Linux were: it runs on a variety of hardware; it provides all the tools a developer may need; natural language processing uses large texts requiring powerful tools; the translation engine uses a large set of rules, hence the migration produced large C/C++ programs and needed powerful tools such as Make and gcc/g++; clients like AltaVista have a large audience and need a robust application on a stable system; and cost.

To these can be added the fact that drivers for newer hardware appear more quickly on Linux than on other platforms, and it uses marginally less resources. Linux provides a homogeneous configuration easy to replicate and is very scalable. Linux also comes with a firewall, sendmail, Apache, modperl and PostgreSQL, all of which are needed for Systran's on-line services (http://www.systranlinks.com/, http://www.systranet.com/). Moreover, environments like GNOME or KDE make it possible to put Linux in the hands of non-programmers as well. This is important because a large number of the Systran

staff are linguists rather than programmers. Finally, POSIX compliance ensures that Systran can port easily to other forms of UNIX.

Systran software is behind most of the automatic translation done in the world. Clients include not only US government agencies and European institutions, but also AltaVista, Microsoft, Apple, Lycos and AOL.

Machine translation is at the confluence of linguistics and computer science. Developing a product is simply translating into computer language all the rules of human language. The main problem is a linguistic one, since you need to start with an accurate description of the languages concerned. There is a description of the source language (the analysis phase) and one of the target language (the synthesis phase).

The code is divided into four parts: 1) the analysis of the source language; 2) the synthesis of the target language; 3) the transfer rules; and 4) the common procedures to all translation engines, i.e., memory management, command-line management, dictionary lookup procedures, filters, pre-processing, post-processing, etc.

The dictionaries used are very specific; they do not only include the translation of the words (i.e., manger = to eat) but also syntactic and lexical information, such as “this verb is transitive, it can be used in this specific context in which case it means this.” There are three kinds of dictionaries. The first two are internal, one with simple word stems and the other with complex or idiomatic expressions. The third is external. The latter are created on-demand for a specific customer on a specific theme. Systran also has resource files that contain the flexions for the verbs or the declensions for languages that have them, as well as specific priority rules for the external (customer) dictionary and stylistic indications. All this is coded in C, although the newer extensions generally are coded in C++.

In order to produce the rules, linguists use a graphical interface coded in GTK. The data is stored in an ASCII file that goes through a Perl program to generate the macroinstructions from the data in the code. The dictionaries are built semiautomatically using the rules discovered during the analysis of the language. A unilingual master dictionary is created for each language; terminology is entered, and Systran's tools automatically add the relevant linguistic information on the base of tables. For example, “automatically” would be recognized as an adverb because it ends in “ally”. Bilingual dictionaries are then built by creating a simple double-entry list, which will then retrieve the relevant syntactic information from the master unilingual dictionary.

It is only at the last stage that the dictionaries are compiled into binary format in order to increase processing speed at runtime. When you clicked on the Translate button on AltaVista, you probably never thought the process behind it was so complex!

Systran is preparing a free Linux release with all the features of the Systran Personal Windows edition.

—Thunus F., Director of Systran Luxembourg

NASA's JPL Builds War Game Simulator On Linux

The Jet Propulsion Laboratory (JPL) of Pasadena, California is one of the space program's major players. Managed for NASA by the California Institute of Technology, JPL is the lead US center for robotic exploration of the solar system and its spacecraft have visited all known planets except Pluto. In addition to its work for NASA, JPL conducts research and development projects for a variety of federal agencies. One such project, the Corps Battle Simulation (CBS) recently made the transition from VAX to Red Hat Linux Version 7.0, resulting in a substantial increase in performance at considerably reduced cost.

CBS has been used to train army officers in battle tactics for over 15 years. Previously, it ran on VAX's most powerful computer, a $100,000-plus 7800-series machine. However, due to the steadily increasing intelligence and the addition of new features, CBS reached its limitations on VAX. This made further innovation a struggle and threatened to render the battle simulator obsolete within a few years. As a result, the US Army's Simulation, Training, and Instrumentation Command (STRICOM), in Orlando, Florida asked JPL to port the software to Linux in order to increase functionality while cutting cost.

After spending a man-year reconfiguring CBS source code, then recompiling, testing and debugging, the team benchmarked the system running on Linux with rewarding results. “By porting CBS from VAX to Linux, we have achieved far better performance at a much reduced cost and have lots of extra capacity,” says Jay Braun, a simulation software technologist at JPL.

The additional capacity of Linux gives the CBS system more room to expand. Terrain elevation, for instance, can now be modeled at a very detailed level. Previously, attempting complex line of sight calculations severely taxed VAX capabilities. Now, high-fidelity maps are available on Linux that make simulations more realistic, increasing the accuracy of the battle scenarios.

CBS is running on a $4,000 PC with a 1.2 gigahertz AMD Athlon processor. This Linux machine runs the largest CBS exercise almost four times faster than the most powerful VAX without sacrificing anything in model fidelity. Using the VAX,

fidelity had to be reduced in order to allow a simulation to progress at a one-to-one game ratio, i.e., a virtual minute in the simulation requires a real minute of execution time. Under Linux, however, one-to-one scenarios can be achieved at the highest quality levels available.

JPL has also made adjustments so that CBS has a 20-second save time for the largest exercises and three seconds for small exercises. This is an order of magnitude faster than on the old VAX system. Under Linux the application can now represent almost 3GB of virtual address space for each simulation. “That's a big image!” says Braun. “Our model has plenty of features that are pushing the limits of Linux.”

JPL will deliver the ported software in June of 2001. Braun predicts that in the near future, the system will further advance to a two-processor machine that can support additional simulations. JPL is now shifting over to Red Hat Linux 7.1 with the new 2.4 kernel.

Fine Print

No text selections can be copied from this book to the clipboard....No printing is permitted on this book.... This book cannot be lent or given to someone else....This book cannot be given to someone else....This book cannot be read aloud.

—From the “permissions” that accompany *Alice in Wonderland*, as published by Adobe in its downloadable .pdf format. *Alice in Wonderland*, written by Lewis Carroll in 1865, has long since passed into the public domain.

Obviously some protection of copyrighted material will, and should be, built into code. But the power to control perfectly the use of copyrighted material should not. The key will be to find a balance. And when companies like Adobe clearly signal that their effort is to find a balance, they deserve the benefit of the doubt.

—Lawrence Lessig, *The Industry Standard*, March 27, 2001

LJ Index—June 2001

1. Development cost, in billions, of the Iridium satellite-based mobile phone system: 5
2. Sale price, in millions, of Iridium, after bankruptcy: 25
3. Number of Iridium satellites that would have been force-burned back to Earth if the system hadn't been sold: 60

4. Rejected sum, in billions, offered to record companies by Napster for allowing copyrighted works to be exchanged on the service: 1
5. Estimated cost of streaming 90 minutes of music to one listener: $81 US per day
6. Estimated delivery costs for the same on a peer-to-peer subscriber basis: $15 US per day
7. Percentage of time spent on-line “accounted for” by AOL-TimeWarner: 32.7
8. Percentage of “at-home penetration” by AOL-Time Warner: 74.8
9. Page-view percentage devoted to the 1,000 most popular web sites in June 2000: 53
10. Page-view percentage devoted to the 1,000 most popular web sites in January 2001: 48
11. PDA sales, in millions, in 2000: 9.39
12. Projected PDA sales, in millions, in 2004: 33.7
13. Approximate percentage of global PDA sales Sharp hopes to capture with its new Linux-based PDAs: 50
14. Sharp's global sales goal in millions for the year ending 2002: 1
15. Number of Java-based programs Sharp hopes to see running on its Linux-based PDA by October 2002: 10,000
16. Sharp's estimate of the number of active programmers for the Linux PDA platform: 100,000
17. Sharp's estimate of the number of Microsoft PDA programmers: 50,000

Sources:

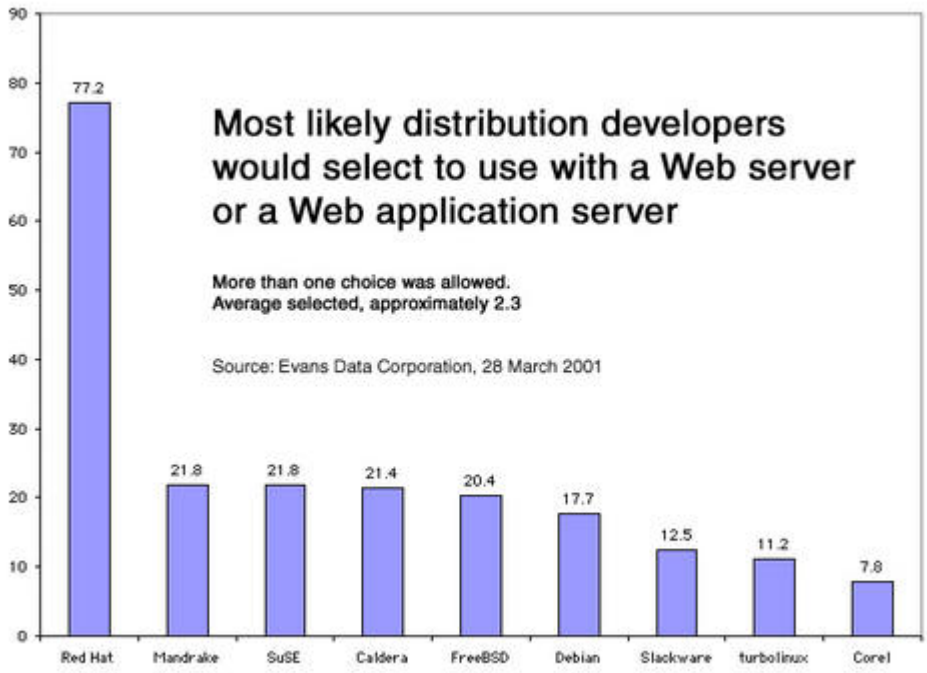
- 1-3: Hoovers
- 4-6: ZDNet
- 7-8: Mediametrix
- 9-10: Industry Standard from Alexa Internet, March 2001
- 11-12: Gartner Group
- 13-17: CNET

Seeing Red

At the NBA's first three-point shooting contest, Larry Bird looked at his opponents in the locker room and said, “Who's playing for second?” That's the main question when it comes to Linux distros. Red Hat has had the Larry Bird position for years now, and about all that's changed is who comes after #1.

Recently Evans Data Corporation of Santa Cruz, California asked 300 Linux developers which distributions they would select for a web server or a web

application server. The obvious answer was Red Hat. Coming in second were SuSE and Mandrake, each with 21.8%. As the chart shows, though, the question for developers really is “Who would you select in addition to Red Hat?” The average number of additional choices was 1.3 (for a total of 2.3 choices). Caldera, Debian and FreeBSD weren't far behind SuSE and Mandrake.



The survey's table of contents is on the Web at Evan's site (www.evansdata.com/Linux01TOC.htm).

They Said It

Between truth and the search for truth, I opt for the second.

—Bernard Berenson

If you tell the truth, you don't have to remember anything.

—Mark Twain

Honesty is the best policy. If you can fake that, you've got it made.

—George Burns

A closed mouth gathers no foot.

—Simon Murcott

Nearly all men can stand adversity, but if you want to test a man's character, give him power.

—Abraham Lincoln

The strategic goal here is getting Windows CE standards into every device we can. We don't have to make money over the next few years. We didn't make money on MS-DOS in its first release. If you can get into this market at $10, take it.

—Bill Gates

Do we have a way for people who host web sites on Linux to build on [.NET]? Yes, we do. That's not to say our overall strategy is not to get those web sites over to Windows, but we will provide a way for those Linux servers to use .NET.

—Steve Ballmer

I'm not one of those who think Bill Gates is the devil. I simply suspect that if Microsoft ever met up with the devil, it wouldn't need an interpreter.

—Nick Petreley

Storage is like eating. You can eat cheaper, but you can't not eat.

—Colin Ferenbach, on prospects for the storage firm EMC

It's a great year for entrepreneurs. The problem is that VCs haven't been investing in entrepreneurs, they've been investing in figureheads with no technology.

—Dave Winer

The only thing you can't do with open-source software is make monopoly profits.

—Jeremy Allison

The first thing that happened after we opened sourced InterBase was customers wanted to know how much it cost. The most important new feature, after open sourcing, was the price tag.

—Ted Shelton

At least, thanks to open source, the technology doesn't die with the company.

—Deirdre Saoirse

Hey, for the price of a distribution, you can have a year of *Linux Journal*.

—Evil Bastard, on OpenSourceRadio

People who whine that Linux user groups exist to “help” people invariably use proprietary mailers.

—Rick Moen

For every traction there is an equal and opposite retraction.

—Doc Searls

What is wanted is not the will to believe, but the will to find out, which is the exact opposite.

—Bertrand Russell

Nobody can jump to confusions faster than the Linux community.

—Arne Flones

The right way to do things is not to try to persuade people you're right but to challenge them to think it through for themselves.

—Noam Chomsky

all your apt-get arebelong to us. dist-upgradenow for great honour

—Debian Haiku by Marc Merlin

Think of It as an Ego Auction

So, how often do people search for your name on Google? To find out, we made an ad on Google's AdWords page, then asked Google to estimate how many times a month we would have to pay to run it when users searched for each of the following names:

Larry Augustin: 0

Chris DiBona: 0

Phil Hughes: 0

Rob Malda: 0

Don Marti: 4,000

Rick Moen: 0

Bruce Perens: 0

Eric Raymond: 4,000

Doc Searls: 0

Richard Stallman: 0

Linus Torvalds: 1,300

Richard Vernon: 0

Bob Young: 0

And, of course, we tried it for operating systems too:

Linux: 4,284,200

Windows: 5,653,800

UNIX: 872,900

There is no charge to get estimates. Try it yourself at http://adwords.google.com/. There is no charge to get estimates.

Things Women Hear at Linux Events

I have a few questions about Linux in general. Is there somebody here that can answer a couple questions for me?

Are you here by yourself?

You use Linux? Well, good for you!

After you're done helping her, can you answer a question for me? (Spoken over the head of a woman volunteer at an installfest, to the person she's teaching to install Linux.)

We only have men's extra-large shirts...but here you go, you can wear it as a nightshirt.

We need some more coffee over here.

Are you in marketing?

—Don Marti

Archive Index Issue Table of Contents

Advanced search

Has Linux Become a Flop?

Phil Hughes

Issue #86, June 2001

What's happening with Linux?

Let me get the punchline out of the way: No, it hasn't!

There, now we can get down to business. Recently, we have seen Linux-related stocks hit all-time lows, an assortment of Linux magazines die, Linux companies fold or cut back significantly and other companies pull out of the Linux market.

What does this mean?

Well, rather than using my crystal ball, I am going to use our eight years of experience within the Linux community and some logic. I hope this will be almost as reliable as the old crystal ball.

When Linus decided the world needed Linux, his decision was based on the fact that he thought he could create something that was needed, a work that offered something new to people around the world. Time has shown he was right.

In 1993, when I decided the world needed *Linux Journal*, it was because I believed in what Linus was doing and felt there was a need to offer a professional journal to help spread the word. Time and reader feedback has shown me I was right.

So, what's with the current situation? First, take a look outside the Linux community. Sun Microsystems stock that was selling for $120 US per share is now selling for under $40 (actually under $20, but there was a two-for-one split). Apple stock dropped from over $50 to around $25 in one day and, six months later, is still in the low $20s. Cisco did a nosedive from over $40 to under $20 in under two months. I could go on.

Now, while the stock market hasn't had a real winner lately, not everything dropped 50% or more. What happened is investors got scared and took money out of what they saw as speculative investments and put it into more secure investments. Anything associated with the dot-com racket was considered seriously speculative and was hit extremely hard.

Don't believe me? Look at IBM. While everything else in technology was going down, IBM was fine. Why? Because it is the big, stable one.

Now, on the other end of the spectrum, let's look at Linux, the new kid on the block. Microsoft says Linux is bad news. All the venture capitalists who jumped on the Linux bandwagon backed off. All the companies that decided they should expand into the Linux space (that includes hardware, software and publishing) realized they were out on a speculative limb and backed down. Thus, Linux is the current victim of short-term desire for big profits.

This reminds me of a phone conversation I had back in the mid-1980s. SSC used to teach a bunch of UNIX-related classes; that is, a short class for managers, a nontechnical class, a class for programmers and a class in C programming. I was talking to a person about the C class. When he found out it contained a lot of information about the UNIX C compiler, UNIX-related libraries, the **make** command and other software unique to UNIX, he told me we could get a lot more students if we took all that stuff out so MS-DOS users would want to attend. I explained to him that SSC's focus was documentation and training related to UNIX, not computers in general. He thought I was crazy.

Well, maybe I was, and maybe I was crazy again when I decided *Linux Journal* was needed, but at least I feel good about my insanity. Today, you see the people who got into Linux for the money leaving town. Unfortunately, that means some good people who believed in Linux but got into the venture capital game are getting hurt. However, that doesn't discount the value of Linux.

In fact, let's talk about the value of Linux. If this is a real economic downturn rather than just money shifting around, Linux becomes more valuable. For example, an article in the May/June issue of our sister publication, *Embedded Linux Journal*, is about the US Postal Service using 7,200 Linux-based computers to read zip codes and write bar codes on envelopes. That's 7,200 computers that didn't need licensing fees paid for a proprietary operating system.

That's just one example. Look at the number of web servers in the world. Every Linux-based web server is one less software license. All this use of Linux shifts money from unnecessary software licenses to other areas. Those areas could include hardware, salaries or just plain savings.

Linux use continues to grow. If you don't believe that, start counting web servers and embedded systems. Sure, the majority of computer desktops don't run Linux, but that is actually the area where Linux has the least penetration. Linux is doing fine. It will continue to grow in use and value. It's too bad some Linux believers got off track thinking the end goal was money rather than world domination. This little shake-up should help us get back on track.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #86, June 2001

Our experts answer your technical questions.

PPP on a PCMCIA Modem

My Dell arrived with factory-installed Red Hat 6.2, Kernel 2.2.14-6.1.1, and a factory-installed PCMCIA modem. All worked fine until one day, I turned on the PC and at bootup I was told:

```
Bringing up interface ppp0 FAILED
```

Somewhat later during bootup, this message appeared:

```
Couldn't configure serial #1 (port=760, irq=3):
  device already open
serial_cs: register_serial() at 0x2f8 IRQ 3 failed
```

This modem card works fine in my Gateway Windows box. Dell refuses to help. —Steve Lohr, steve@azstarnet.com

It's possible you have a program running that is tying up your serial port. Two common culprits are GPM, the text mouse dæmon, and one of several UPS monitoring dæmons. Use the **ps ax | less** command to look for such culprits. If you are still having problems identifying the culprit, try booting into single-user mode to shut down all unnecessary programs. Then use minicom to access the port directly, and verify that it can see your modem. —Chad Robinson, crobinson@rfgonline.com

Perhaps a lock still exists from a previous session not ending normally. See if there are any old lock files in /var/lock, such as LCK..modem or LCK..ttyS1. If there are, just delete the files and reboot. —Keith Trollope, keith@wishing-well.demon.co.uk

I Have No Nameserver and I Must nslookup

I am setting up a Linux machine that will masquerade my workstations and provide port-forwarding to my web/mail server. My question is, do I need an internal DNS server on my Linux machine to serve up DNS requests in order to browse the Web on my internal workstations? I would rather use the hosts file to define name-to-IP translations of my internal network. —Brandon Zumwalt, bzumwalt@seventhview.com

You do not need a DNS server on your Linux box, but you also cannot use the hosts file to perform this lookup. The hosts file is used by services running on your Linux box itself. Your first option is to configure your internal systems to use the DNS servers provided by your ISP. Once you have masquerading set up properly, your workstations will be able to access them and resolve web addresses without further work. Alternatively, you can set up a DNS server on the Linux box that will cache workstation requests. If you want to simplify this configuration, use the forward-only sample configuration for BIND. That puts the server into caching-only mode, with no local tables of its own. —Chad Robinson, crobinson@rfgonline.com

Check your distribution's site for BIND security updates. Old versions of BIND, like the one that's probably on your distribution CD-ROM, are subject to automated attacks. Or run—Don Marti info@linuxjournal.com

X and Sound over the Net

Is there a way I can have an X window appear on more than one X server (display)? I may want to do it (perhaps read-only) for, say, DVD output to displays throughout my home.

Separately, when I run an X application from a remote machine, how can I get the sound to follow and output on my local sound device? I only seem to be able to get sound out of a configured sound card for the host that owns the application. Similar to the above, can this be redirected to more than one address (back to my DVD playing in every room fantasy)? —Matthew Holmy, mholmy@yahoo.com

Xmx will allow you to display a normal X application on several X displays, http://www.cs.brown.edu/software/xmx/. However, for display speed reasons, some applications (like DVD players) bypass most of the X server when they write to your video card. Unless they have a slow, normal display mode, you can't do a remote display. You cannot send full framerate uncompressed video over Ethernet; it's too much data. You should, however, be able to rip a DVD on one machine, send it over the network and play it from disk on a different one. The network audio system lets you play sound over the network, radscan.com/

nas.html. Another program you can look at is located at http://rplay.doit.org/. —Marc Merlin, marc_bts@valinux.com

make Just One Module?

I recently acquired a USB scanner (Epson Perfection 1640SU) and was successful in getting it to work with my Linux system (kernel 2.2.16-22). However, I need to modify one of the timeout parameters in scanner.h in the kernel source. How do I recompile this module to get scanner.o without having to recompile the entire kernel? Do I make a backup of /lib/modules/2.2.16 before I do this, and can it be safely restored in case of error? —Jin, j.r.ong@ieee.org

After modifying the file, simply type **make (target)**, e.g., **make bzImage**, from your /usr/src/linux or similar directory. This will recompile only the necessary files. The Makefile set for Linux has been altered to always compile certain files, such as main.c, but only a few files fit that category. All of the drivers should be skipped except the one you've changed. If you've changed only the .h file, the **make** program may not recompile your driver. Try **touch scanner.c** to simulate having made changes to the file. In addition, you can make your life a lot easier by simply using a module instead, in which case you would not need to recompile the kernel, only rerun—Chad Robinson, crobinson@rfgonline.com

fsck without Reboot

I'm building a series of servers for a friend of mine, and they must stay on 24 hours a day, 7 days a week, 52 weeks a year (well, if they don't burn, obviously).

Is there a way to automatically recheck a mounted filesystem? —Franco Favento dei Favento da Trieste, f.favento@ieee.org

Switch to reiserfs, a journaling filesystem for Linux. Even without it, I have had Linux servers running for years without filesystem glitches, but reiserfs is more reliable. But if you really want to do a filesystem check, boot using a very minimal (better still, ramdisk) root filesystem, and store on it only those executables needed to run a minimal system. That will allow you to unmount the filesystems you actually need to check. —Chad Robinson, crobinson@rfgonline.com

Only a read-only check can be done on a mounted filesystem. It cannot be repaired:

```
fsck.ext2 -fn some-device
```

If it detects errors, you can plan downtime to repair the filesystem. —Keith Trollope, keith@wishing-well.demon.co.uk

Files Larger than 2GB

I am happy to report that my employers are moving to Linux in a big way for seismic data crunching, and they've given me a nice symmetric multipenguin. However, I routinely use files greater than 2GB. I've got reiserfs and a 2.4.2 kernel on the RH 7.0 installation now, but still can't deal with files greater than 2 —Adam Cherrett, adam.cherrett@elfgrc.co.uk

You should look at http://www.suse.de/~aj/linux_lfs.html. In addition to kernel and glibc support, you need to do one of the following in your programs: 1) Compile your programs with gcc -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE. This forces all file access calls to use the 64-bit variants. Several types change also, e.g., off_t becomes off64_t. It's therefore important to always use the correct types and to not use, for example, int instead of off_t. 2) Define _LARGEFILE_SOURCE and _LARGEFILE64_SOURCE. With these defines you can use the LFS functions like open64 directly. 3) Use the O_LARGEFILE flag with open to operate on large files. —Marc Merlin, marc_bts@valinux.com

Finding Memory Hogs

I have recently installed Linux on my PC with 256MB RAM. I wanted to use it as my experimental system with an Oracle server and for some Java development.

To my surprise, the system was striving for resources, especially memory. So I booted it fresh, and without an X session I checked for memory usage with **free**. It was showing that about 110MB was used. How do I decide what processes are using how much memory? —Dhimant Patel, a24z57k@runbot.com

Be sure to note the “-/+ buffers” line when running the **free** command. Linux will automatically use available RAM to buffer I/O requests, and this memory will be freed for program use as necessary. Your primary concern should be the “used” indicator for your Swap space. It should be small—less than a few megabytes. You can use the **ps aux | less** command to examine the memory usage of each running process. Only the resident set size (RSS) value should be important here, but be aware that the memory indicated is not necessarily used as-is by each process. That discrepancy has to do with the fact that **ps** will show all memory used, even by shared libraries, although they are loaded only once for all processes that need them. —Chad Robinson, crobinson@rfgonline.com

make zdisk Makes SuSE Box zdead

I compiled kernel 2.2.16 from SuSE using **make zdisk**. When I boot my system, the progression dot goes to the end of the screen and bombs out with the error mesage “out of memory”. —Eskinder Mesfin, amesfin@uhc.com

You need to compile with **make bzdisk** to solve that problem; it shuffles memory around in different ways to make bigger kernels work. —Marc Merlin, marc_bts@valinux.com

Matthew 9:17

I made some backup tapes using **ftape** (HP/Colorado Travan 1). I could read them fine on Red Hat 5.2, but on later releases (e.g., 6.2, 7.0) it acts as if nothing is on the tape. —Jim Haynes, jhaynes@alumni.uark.edu

In the newer versions of ftape, a fixed block size on the tape is used rather than the variable size previously used. To change the block size to a variable size, type:

```
mt -d /dev/qft0 setblk 0
```

Information on the old and new versions can be found on the >tape home page, http://www.instmath.rwth-aachen.de/~heine/ftape/. —Keith Trollope, keith@wishing-well.demon.co.uk

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #86, June 2001

Agenda VR3 PDA, Game Channel Solution, Smoothwall v. 0.9.8 and more.

Agenda VR3 PDA

Agenda Computing unveiled the Agenda VR3 PDA during Spring COMDEX in April 2001. The VR3 runs Agenda Linux, an embedded operating system designed to run compatible handheld and e-mobility devices. In addition, the VR3 is powered by an NEC VR4181 66MHz, 32-bit processor and comes with 8MB RAM and 16MB of Flash memory. CIR circuitry and software are standard, allowing the unit to act as a remote control for household appliances. The VR3 can also send memos or messages to a printer or other device by wireless infrared transfer.

Contact: Agenda Computing, #368, 4521 Campus Drive, Irvine, California 92612, 888-741-8181 (toll-free), sales@agendacomputing.com, http://www.agendacomputing.com/.

Game Channel Solution

Not a Number, creator of the Blender 3-D software, announced the Game Channel Solution (GCS) for providing distributed, multiplayer games services on-line. GCS addresses the gaming community, including developers, ISPs, resellers and consumers, allowing them to choose the features suited to them. For a flat fee per subscriber, GCS provides hardware and software implementation, connectivity and bandwidth, billing and records of customer use, content management and distribution, and multiplayer, real-time content. GCS offers a wide range of game titles and licenses, as well as original content from PC developers and unique Blender titles that are available for a variety of supported platforms.

Contact: Not a Number, van Eeghenstraat 84, 1071 GK Amsterdam, The Netherlands, +31-(0)20-3058250, http://www.blender.nl/.

Smoothwall v. 0.9.8

The new version of Smoothwall security device software is available for free download under the GNU GPL. Smoothwall allows PC users in home, SoHo and office environments to build a web-managed, secure internet router that provides a secure network internet connection. New additions to this version of Smoothwall include automatic probing and setup for ISDN devices, multiple internet devices (allowing the router to be used for web-site hosting), IPSEC VPN capabilities and support for ADSL and cable users. The download is available at http://www.smoothwall.org/dyn/get/download/html/.

Contact: Scorpio Network Technologies, Ltd., Open House, 3 Thames Court, Richfield Avenue, Reading, RG1 8EQ, United Kingdom, +44-0-118-956-6116, http://www.smoothwall.org/.

Kapital for KDE

Kapital, a product of the Kompany.com, is a personal finance package for KDE. Features include a register for various types of transactions, a calendar and “bill tracker” alarm for scheduling payments, check and report printing, searching and on-line reporting capabilities with charts and graphs, predefined and user-added categories, a new account wizard and import/export features for Quicken. Kapital also allows for on-line banking and budget tracking and the ability to manage multiple account types.

Contact: theKompany.com, Inc., PO Box 80265, Rancho Margarita, California 92688, 949-713-3276, info@thekompany.com, http://www.thekompany.com/.

StoreSense for VA Linux

The flagship e-commerce solution from Kurant Corporation, StoreSense, is now available for VA Linux's 1U and 2U server lines. StoreSense offers a suite of e-commerce tools and services, enabling businesses to build, manage and maintain internet storefronts. Using StoreSense, ISPs can deliver customized services to business customers, ranging from entry-level catalog capabilities to supply chain management and wireless shopping. StoreSense is offered on a subscription basis and is compatible with Linux, Windows NT, Windows 2000, Solaris and Cobalt's RaQ server platform.

Contact: Kurant Corporation, 32 Cleveland Street, San Francisco, California 94103-4014, 916-984-5400, info@kurant.com, http://www.kurant.com/.

Maya 3-D Software

Alias|Wavefront's line of Maya 3-D software products has been ported for use with Red Hat Linux (6.2 and higher) to enable development of games, films, visual effects and all types of animation. The components of the Maya line, including Maya Builder, Maya Complete and Maya Unlimited, afford users flexibility in choosing hardware and software configurations. Maya is also available for IRIX, Windows NT, Windows 2000 and a Mac OS X version will be released later this year.

Contact: Alias|Wavefront, 210 King Street East, Toronto, Ontario, M5A 1J7, Canada, 800-447-2542 (toll-free), info@aw.sgi.com, http://www.aliaswavefront.com/.

SuperScaler Appliance Servers

e-Appliance, a manufacturer of high-speed, high-density appliance servers, has added a new family of appliance servers called SuperScalers (SS). The SuperScaler 600 is a 1U 19-inch rackmount containing four independent, hot-swappable servers. Each server contains its own processor, memory, disk drive, power supply and Ethernet connection. The SS1000 features two independent, dual SMP servers and fiber channel technology; it is designed for intensive high-speed processing applications like streaming media. The SS500 contains one dual SMP server and has optional fibre channel connectivity. It is geared toward applications that need a balance of processing power and network performance.

Contact: e-Appliance Corporation, 3052 Bunker Hill Lane, Suite 101, Santa Clara, California 95054, 408-980-1990, operators@eappliancecorp.com, http://www.eappliancecorp.com/.

Archive Index Issue Table of Contents

Advanced search