Advanced search

Linux Journal Issue #83/March 2001

Untangling MPEG Source—Chiariglione Speaks
LINUX JOURNAL
Embedded GUI
Greg Haerr Reveals his Plans for Microwindows Domination
designing DMZ networks
communicating with SOAP
making broken memory work
Becoming a Consultant—
(Or Just Hiring One)
How to be a successful high-tech Linux guru in a Windows world
Making a Living with Open Source

Archive Index

Advanced search

Focus: Consulting

Richard Vernon

Issue #83, March 2001

This month's feature articles are addressed both to those who belong to, or are considering starting, a one-person or small consulting business, and those looking to hire a consultant.

This month's feature section is devoted to a less technical subject—consulting. The articles are addressed both to those who belong to, or are considering starting, a one-person or small consulting business, and those looking to hire a consultant. As the use of Linux proliferates it presents a potential hotbed of opportunity for those Linux professionals with the skills to sell their experience.

December's issue on system administration mentioned the frustrations of being locked into a working relationship with nontechnical people. Consulting, however, involves the potential frustration of not being in a working relationship with anyone at all and the possibility of calling it quits and going back to work for “the man”. On the other hand, being self-employed has its advantages, not the least of which is freedom and flexibility.

A consultant is a bit of a mercenary (Doc, forgive the bellicose metaphor), a knight errant or a “Have gun, will travel” whose metaphorical gun is experience and ideas. For those attracted by this romantic ideal (and even the disillusioned who work best on their own) becoming a Linux consultant might offer the chance to do what you love and get paid for it. Marty Larsen is a long, time big-time consultant and currently VP of consulting operations for VA Linux Systems, and “Do what you love” is the first item of his article in which he explains five requirements for the successful consultant.

Part of the value of an outside consultant is that they provide a fresh perspective to solving a company's problems. Consider two examples from literature. Shakespeare's Polonius, “consultant” to the Danish king in *Hamlet*, is definitely part of what is rotten and stagnant in Denmark. His entrenchment in his position blinds him, and his councils are nothing more than worthless platitudes. He and those he consults end up dead (hey, it's a tragedy). On the other hand, Joseph of colorful-coat fame gets sold into Egypt with a novel outlook, coming from a different land, different culture and a different god. He starts out small (consulting fellow prisoners), gets one good reference and in no time he's consulting the pharaoh and practically ruling Egypt (of course, having the gift of prophecy doesn't hurt).

Glen Otero, who runs his own small consulting firm, Linux Prophet, points out that while Linux, being “a fresh perspective”, may represent great opportunities for consulting, one can't expect to get a job based solely on its merits, some of which, while endearing to the Open Source community, count for very little with many companies. Based on his experiences, Glen writes that Linux is simply a tool, and a good consultant must know that tool intimately enough to show exactly how it delivers the best performance.

Joshua Drake, another independent entrepreneur, gives some attention to the other side of the consulting equation—the client. He reveals some important considerations in finding a good consultant as well as some likely places to look.

While not found in the Feature section, but in its usual spot of honor, Marcel Gagné's “Cooking with Linux” this month deals with applications such as TimeSheet and BANAL that aid the Linux consultant with those arduous tasks of time tracking and bookkeeping. And don't forget to give Ralph Krause's review of *The Computer Consultant's Guide*, 2nd Edition a read.

—Richard Vernon, Editor in Chief

Archive Index Issue Table of Contents

Advanced search

How to Be a Successful High-Tech Consultant

Marty Larsen

Issue #83, March 2001

Number one requirement of the successful consultant: do what you love.

There are many means by which we choose to measure what is success. In my early days of computer-industry consulting, I was happy and successful because I was independent, busy and doing what I enjoyed. My primary goal was to make a living as a computer engineer and not work for a defense company. Although the economy was slow, I was successful because of a simple principle I have always stuck with: do what you love.

However, as time went on my success had limits. In the early days, I was a “body for hire”. I went from job to job, and when I had some free time I did my “administrative work”. The not-so-fun paperwork, billings, etc., always came last and, even then, only when I was forced. This model was cool but not very scalable. My priorities began to change, and my need for a more stable, steady income stream became apparent. I had to start exploring the more advanced consulting model.

When I examined my operation I found that life was good when I was doing my thing for the customers. My pain was in between jobs, when I had to collect bills and actively market my skills in the ever-present search for that lucrative, long-term and very cool gig. In business terms, I was at the sawtooth growth point for which the current mode of operation required additional investment for continued growth. I had the choice of hiring some help or hooking up with some outfit that would feed me leads; in return I would “fill out” their service offerings. My new expanded operating model evolved into two cool gigs. I hooked up with a small company hocking computers, Advance Engineering Labs, and I sold “solutions”. The solution lead to the sale of hardware. I got paid for designing and implementing the solution that solved a business problem for AEL's customers, and AEL got paid for the sale of the hardware. Together we offered complete solutions and a single point of contact for the customer. This arrangement worked out quite well, and we were fairly unique in the early days of desktop-computing environments.

Part two of my mode of operations was part-time teaching of digital-design engineering at Heald College. I also credit my teaching time as excellent presentation training—a valued skill as we move up the ladder in the consulting world. Once again I felt I was the “successful” consultant.

However, as my customers grew, so did their needs and their level of engineering complexity. The increase in demand forced me to give up teaching and concentrate full-time on the consultative role I performed.

Item number two of requirements for successful consulting: continuing education, both formal and informal via industry seminars and trade rags.

The shelf life of consultants that do not continue to update their education is about the same as a cup of yogurt. I cannot express the importance of continued education. As consulting gigs grow in size and sophistication so must your resumé. Customers want to know how you keep up as well as what you have done.

Item number three: a highly referenceable, in depth library of information available at one's fingertips.

As my number of consulting gigs increased I began to amass quite a collection of programs, diagrams, documents, tools and experience. Reinventing the same stuff is a sin. The independent consultant with any hope of being self-supporting must have an organized library of information available at a moment's notice. As the consulting gigs grow in complexity so must your ability to communicate complex solutions in nontechnical ways. A solid library of information, processes, techniques, programs and tools can “jump-start” a consulting engagement, and develop confidence between you and the customer. Every customer likes to know they are getting their money's worth.

Item number four of the successful consultant: require an assessment. Deliver a fair, honest, objective assessment to your customer.

If we are to be more than a “body for hire”, then we need to be part sales, psychologist, engineer, accountant, business person and so on. We must understand clearly the needs of the customer, the problems they are toiling with and the urgency of their situation. The best way to gain the confidence of the customer is through a proven-tested-assessment process designed to clarify and articulate the customer's need. In other words, before you jump in with your “solution offering” you must assure the client you are completely aware of their “unique” and special situation. Insist on an assessment! This can be as simple as looking at the code and as complex as documenting and diagramming the customer's entire business process. The final product of an assessment should be impartial in nature and of value to the customer. The successful consultant does not go into an engagement blind. We would be doing the customer a huge disservice if we did not carefully assess the situation, report our findings, and then collectively design a solution. If we maintain an objective approach, and if we are as complete as possible, the assessment should be of value to the customer and, therefore, a paid engagement. All our customers are offered a “free” assessment by the other guys. There are no free lunches! Your customers will get what they pay for, and almost all of the them will know this. If they do not, you may want to rethink the engagement. People not willing to do their homework most often are not satisfied with the final results.

Item number five for successful consulting: finish big.

Let's say you are a consultant doing what you love, continuing your education, amassing your library and performing valuable assessments. What is next? Do the work right? Actually, performing the work is the easy part as long as it is something you love. The next most important thing you can do to maintain your success as a consultant is to “finish big”. Boxers may get away with poor training, starting slow and performing poorly for nine rounds, but if they knock out the opponent in the tenth they win. On the other hand, if the boxer trains well, studies his opponent, and kicks butt for nine rounds but drops his guard for one second in the tenth and gets knocked out, then he is remembered as a loser. The same goes for the consultant. Maintain your presence and build your business through references and repeat business, and then give your customers something to remember you with. Do a final presentation of your work, summarize your accomplishments throughout the engagement, present your final report and include a copy of all the kudos the client wrote to you while you were engaged. Do whatever you can to finish big. It is your last and best chance to build more business out of a successful consulting engagement. Even if the engagement was not successful, a final presentation of the project, your work and the issues is viewed as a professional thing to do and will often get you a second chance.

Marty Larsen has been doing IT-related consulting for over 15 years. He is currently the VP of consulting operations for VA Linux Systems Inc. Marty has done extensive IT consulting for the oil and gas as well as high-tech industries. His past positions include director of oil, gas and communications consulting at EDS and manager of Global Capacity Planning for Intel Corporation's IOS data centers. He has a BSEE from Columbus University.

Linux Prophet

Glen Otero

Issue #83, March 2001

Linux consulting is not really about Linux.

I run a consulting firm by the name of Linux Prophet in San Diego, California. Actually, I am Linux Prophet. I design and build Beowulf-type clusters for biotechnology and academic clients. *Linux Journal* asked me to share some perspectives and advice that I've gleaned from being a one-man Linux show. I was more than happy to oblige.

People interested in building a consulting company around Linux should come to grips with the fact that, to the rest of the world, Linux is not a no-brainer. Not yet. What I mean by that is, believing that you will be given a contract “because Linux is cool” is simply boorish. You need to be able to explain in painstaking detail why a client should use Linux. Educating people about Linux, and open-source software in general, is an important part of my job. And the Linux edification, no sermons please, often isn't enough to land a contract. “Why is that?” you ask.

It's because you won't be preaching to the choir when it comes to landing a contract. The decision makers won't be Linux zealots or even know what Linux is. And even if the people you are negotiating with have heard of Linux, they won't care about price points, monopolies, the morality of software ownership or world domination. They will demand high performance and reliable solutions. Period.

Why don't they care about price? Well, for one thing, they are used to being gouged by other proprietary frameworks. Secondly, serious players know that you get what you pay for and that there is no such thing as a free lunch. So harping on the fact that Linux is free may send up some red flags to potential buyers and/or lead you to underbid for a job. Emphasizing free as in freedom, as opposed to no- or low-cost, can potentially avoid this. Monopolies? Last time I checked, the judgments and appeals process hasn't prevented people from purchasing or upgrading the most popular desktop operating-system software. Call Microsoft what you will, they are entrenched in most institutions and won't be disappearing overnight. Morality? Regardless of who your potential client is, they won't see proprietary, or any other software, as evil incarnate—except maybe spamming software. Open-source zealots aside, I have yet to meet anyone that thinks free software is a moral imperative.

What's my point? My point is that nobody hires me for the sake of Linux. They hire me because I do a great job. I use the best tools available to deliver a solution, open source or not. In Beowulf-type clustering, the best tool is Linux.

If this sounds like a jaded perspective, maybe some backstory will help explain.

My history with Linux began a few years ago with my foray into bioinformatics. Bioinformatics, aka computational biology, is essentially the extraction and organization of knowledge from the reams of protein, genomic and other available genetic data. When I decided to leave the wet-lab bench and concentrate on a career in computational biology, I pondered on the toolset I would need to transition into bioinformatics. That's when I was introduced to the Open-Source software movement and some corresponding tools like Perl and Linux. Perl has done all kinds of heavy lifting in several genome projects. I also realized that the questions I wanted to ask were going to require serious computational resources to answer—enter parallel programming. Clusters were beginning to gain popularity at the time, and the OS of choice for scientific, clustered computing was Linux. I slowly developed the vision that bioinformatics software of all sorts, running on Linux clusters, was going to be pivotal in the age of mining the human genome. So that's what I set out to do.

Entering the IT arena from a scientific background, and not a computer science or engineering one, has given me a unique perspective on Linux-centric consulting. If there is one bit of advice I can give about Linux consulting, at least in my case, it's that this really is not about Linux. Linux is just a tool. Sure, it's a turbo-charged, nitro-burning, funny-car type of tool, but a tool nonetheless. It's not a solution in and of itself. When you think about it, I'm really a bioinformatics consultant. Parallel programming, the human genome project and computational biology existed, overlapped and got on with one another before Linux hit the scene. And these pursuits are still in no way dependent upon Linux. It's only very recently that Linux clusters have become the premier tool in these fields.

What I'm getting at is that in order for individual Linux consultants to be successful, they need to provide some service, other than distributing Linux propaghandi, and solve specific problems. Flex your expertise as a streaming-media expert, system administrator or Apache stud; do a great job; do it with

Linux and do it better than the next guy implementing that other OS. The idea is that you leverage Linux to make you the best streaming-media expert or Apache guru around. Now this may lead you down a path that ends in Linux omniscience and god of all things Linux. That's great. It's just that no one person can really start there. Start with what you know, and hopefully, probably, Linux can help you do it better. If you don't know your job, or can't address specific problems, then nothing can help you.

Everyone is different, and running your own business isn't for everyone, so I'm not going to recommend a one-size-fits-all solution on how to start your own Linux business, should you decide to do so. But I will say a few things: 1) seriously consider starting out part-time or working for a systems integrator like VA Linux before you take the solo plunge; reading books is in no way a satisfactory preparation for the many intangibles that you will experience when doing business; 2) if you start off on your own, make up your mind from the start as to how you will bill your clients. Even if your decision is to consider two options, limit it to those two options and don't waffle; 3) and if you decide to charge per diem; $1,000/day is a good number to start with. It may sound like a lot of money, but if you're running a solo business properly, you will only have about 20 billable hours a month. And approximately half of what you bill out will go to cover overhead. You also need to take into consideration that you won't always be able to bill that many hours a month, or be working on a contract at all, which leads us to the adage, “I'm not unemployed; I'm a consultant.”

Starting my own business was a great way to scratch my bioinformatics and Linux itches. I know that I'm making contributions to both genomic research and Linux advancement. It's tough going, but nothing worthwhile has been easy for me. Oh, and did I mention that I love every second of it?

Glen Otero has a PhD in Immunology and Microbiology and runs a consulting company called Linux Prophet in San Diego, California. He can be reached at gotero@linuxprophet.com. Surfing, in the ocean that is, is his favorite pastime.

Consultants Revealed

Joshua Drake

Issue #83, March 2001

Joshua gives an honest and thorough profile of the range of consultants.

There is a common misunderstanding in the IT industry that Linux does not offer professional support. I was recently speaking with one of my peers who was having trouble with a Windows NT server that was handling file and print sharing for a medium-sized network. My associate was frustrated because Windows NT does not handle high-bandwidth situations well. If an end user were to drag a very large file over their network, the load on the server would go up considerably.

I asked my peer why he did not use a solution such as Linux with Samba. The answer I received was interesting. It was not “Linux, are you kidding?” or “Linux sucks!” It was “What if it breaks, and frankly, who do I hold liable if it does?” I had to give him a sidelong glance, since one of my company's (http://www.commandprompt.com/) focuses is managed-Linux services.

While talking to other clients, I have received many similar responses. The press has gone to extraordinary efforts to promote Linux. It has also gone to extraordinary efforts to damn Linux. In the end, the press, the key marketing tool of the Linux industry, has missed the entire point. The point is that professional, commercial, Linux-support corporations do exist. Instead the press has concentrated on either Internet-related support resources such as USENET or the Linux Documentation Project (http://www.linuxdoc.org/). They also mention companies like LinuxCare or Red Hat. The problem is you cannot touch, praise or, if necessary, physically strangle Red Hat, LinuxCare or comp.os.linux.networking.

Don't get me wrong. I am playing devil's advocate here. The major firms like Red Hat, VA Linux and LinuxCare have their place in the support marketplace. They are excellent with phone-based support—if you want to pay $325 US per incident to Red Hat. I have news for you: that is more expensive than Microsoft.

LinuxCare rides in at $180 per hour and you specify the cap. Excellent: I get to pay $180 an hour and hope they can fix it. What happens if they don't? At least with Red Hat support is incident-based.

If you are a medium- to large-sized corporation, Red Hat or LinuxCare may be good for you. They offer the type of service that a lot of smaller consultants cannot; they are always available. They can usually offer quicker response times because they have people waiting by the phone just to take the call. They also have a lot of internal resources that they can call upon. If one engineer does not have the answer, the one sitting next to him might.

Price is not the only disadvantage. An engineer cannot *see* what is happening. They must be told, and understanding and accuracy can be lost in the translation. Of course, you could allow the engineer remote access to your machine.

Documentation can be an issue as well. When was the last time you used phone support and, when you were finished, received a detailed summary of what the person did, how much time it took them and what, exactly, was fixed?

To this end, small consulting firms and independent consultants are a good fit for many corporations. If you have 1-100 users, they may be your best solution. The consultant is a sure-fire way to make sure you get your hands on someone if something is broken. Typically, they are local, and sometimes you may even know where they live. There are many consultants who will actually be more knowledgeable than the standard engineers at a place like Red Hat. The consultant will have a tendency to work more directly with machines, have a personal lab and try new things.

The willingness of consultants to experiment can be both good and bad. The good side is the consultant will be familiar with more of the software and solutions that Linux can provide. The bad side is the consultant may find a good solution and charge the customer to install it, when, in actuality, the customer is being charged for the consultant to learn how to install and use the program.

I should stress that this is not usually the case. The business perspective is that you shouldn't have to pay someone to learn how to provide you with the service you need. In technology, however, a customer will often demand a type of technology that the consultant does not know. The business can either pay the consultant to learn and install it or find a consultant that already has knowledge of the technology. It is the consultant's responsibility to be honest with the customer about his or her level and breadth of knowledge.

So, how do you find a consultant? That is a tough one. The largest database of Linux consultants is the Linux Consultants Search and Guide located at http://www.linuxports.com/. It currently has over one thousand entries, with locations ranging from California to Turkey, and you can search via mail code, keyword or other options.

Another possibility is your local Linux user group. You can find a list of them at http://www.linux.com/lug/ [see also Groups of Linux Users Everywhere (GLUE) at http://www.ssc.com/glue/groups/]. Linux User Groups can be an easy way to network and find someone to help you with your Linux machines. If you are trying to locate a Linux consultant, the most difficult part will be finding a good one. Many consultants are extremely independent. They want to remain entirely autonomous; they want to do things their way, and unless you speak geek, they communicate poorly.

The trick is to find a consultant that you can communicate with. If you can find a consultant that communicates well, you will get a lot farther. There is a reason many large consulting firms, Linux or otherwise, have a customer service department or sales people to take your initial calls. Technical people can be very hard people to deal with on a common level. If you are uncomfortable with computers, they can be even more difficult to deal with. To know whether you have found a good consultant is difficult, but there are a couple of things that you can look for:

- A good consultant will be patient without being condescending. They will communicate with you. They will explain everything they have done or will do in complete detail. They will do so more than once, if necessary.
- He or she will provide everything in writing. If they are completing a project for you, large or small, they will provide a project plan. The project plan can be as simple as a half sheet of paper for certain tasks, or it may be ten pages for others. It just needs to be documented. If it is a service call, the consultant will provide more detailed communication as the call progresses. They will help the customer decide what actions to take. When the call is completed, they will provide a written analysis of what they did and how it fixed the problem. The analysis will be repeated on their billing.
- He or she will bid on everything that they do. They will say, “I think it will take two hours to complete this job.” However, it is important for the customer to realize that computers have attitude problems and that most consultants work on a time-and-materials basis. It may take more or less time than the initial estimate. It is the consultant's responsibility to make sure that the customer understands this.
- A good consultant will happily provide a list of references that verify their knowledge and ethics. A note on this: Linux is young, and there are a lot of

new consultants out there just trying to get by. Do not be afraid to use an untried consultant. If you are concerned about their skills, counterbid the price. Ask them if they will come out, without charging you, to assess the problem. Tell them that you will happily pay their full rate, plus the time they spent to assess the problem, if they give you a reasonable assessment and are able to fix the problem. This will allow you to judge their mannerisms, assess how they will perform and give you a general feeling as to whether or not you can work with the consultant.

- A good consultant will not do only Linux. There are a ton of consultants out there who say “We only do Linux.” It is supposed to make them seem like a better consultant, at least technically. The problem is (unless you are a Linux-only company) you will have to deal with many different vendors.

I have a customer that has WTS (Windows NT Terminal Server) 4.0/Citrix, AIX, Linux and SCO. I have to manage all of these machines plus the users that are connecting to them. If I were just a WTS person, I would not be of much use to them. (But yes, I advocate switching all to Linux at every opportunity.) It is very important for the consultant to understand your environment. Do not expect them to be an expert in every environment, but a basic understanding of the environment is important. The ability to perform most common administrative tasks of other environments is also important.

On the other hand, do not expect that the person that has been handling your Windows NT network has a clue about Linux. Linux is a completely different beast than Windows NT. It operates differently and takes a different kind of love to make it work. Linux will run without trouble for years at a time if you have the right person to take care of it. If you get the wrong person, it will be a source of heartache forever. If you are thinking about migrating to Linux and you want to use your existing consultants, make sure that they have actual implementation experience with Linux.

If this is what a good consultant does, you may be asking yourself why *your* consultant is not doing these things. There is a drawback to having a *good* consultant. The truly good consultants, if they are following everything I've described, would have to bill you like a lawyer just to stay alive. When I call my lawyer, even to ask a simple question, I am billed for that phone call. In order for the firm to remain profitable they must bill me. It may only take ten minutes of their time on my end, but it could easily take 30-45 minutes to document the call, give it to the assistant, have the assistant record the call and put the notes of my call in the file for future reference. If you get six of those phone calls a day, that is three hours of time that needs to be billed.

As mentioned previously, the consultant, just like a lawyer, operates on a time-and-materials basis. In order to provide the customer with the level of service just described, the consultant would have to charge more than the standard rate. They would have to take time, and thus money, to create all the documentation. They would have to charge you to communicate, negotiate and sometimes to simply put up with you. The customer may always be right, but it doesn't do the customer any good if the consultant can't get his work done. The more of the consultant's time that the customer takes up, the more expensive the consultant will be. Do not micromanage the consultant; build a relationship of trust with him or her, but make sure he or she sticks to what you agreed upon. It is a very fine line.

Given all of this to consider, how can you tell a good consultant from a bad one? There is much more to a good consultant than technical know-how. There are varying degrees of bad consultants, and I don't think that very many of them are truly bad. Personally, I have a difficult time dealing with “management”. The people wear the ties, read *PC World* and think they have a clue really get on my nerves. Does this make me a bad consultant? No, not if I communicate professionally and am patient with the guy wearing the tie who just read the latest issue of *PC World*.

Again, most consultants aren't bad consultants. The majority of problems that I have seen between consultants and companies were caused by a lack of communication. If you lack good communication skills you will always have difficulties. Communication between the customer and the consultant takes two forms: customer management and consultant management.

A good consultant will have good customer-management skills. Customer management is the ability to communicate with the customer without intimidating, insulting or badgering. It is the ability to resolve issues without becoming defensive. It is the ability to address a customer as an equal but provide boundaries within which they can work with you. The easiest mistake to make as a consultant is to take a call after hours but charge the customer a normal rate. A few may call this customer service, but customers by nature and by good business sense (attempting to save money) will try to take advantage of that. If you do it once, you can mistakenly set a precedenct.

Conversely, a good customer will manage their consultant. The consultant must understand clearly what their role is and what it is the customer would like them to do. If the consultant does not understand these things, you will consistently run into, “But I thought...” or “Wait, you told me....” If you utilize consistent procedures, thorough communication and clearly expressed expectations, you will see the majority of your consulting issues fall by the wayside.

Joshua Drake is an e-commerce and Linux consultant who owns his own company, Command Prompt (http://www.commandprompt.com/). He has been using Linux since the beginning and is the Linux Documentation Project's webmaster. His other projects include the LinuxPorts.com web site and the OpenDocs publishing company.

Deploying the Squid proxy server on Linux

Ian Spare

Issue #83, March 2001

Ian gives an example of the installation, configuration and maintenance of this multi-tentacled invertebrate proxy server.

To provide Internet access for users in the SAS Institute Europe, Middle East and Africa (EMEA), a number of proxy servers have been installed both at the country office level and centrally at SAS European Headquarters in Heidelberg, Germany.

These servers run the Squid proxy server software; this software is available under the GNU general public license. In brief, Squid provides for caching and/or forwarding requests for internet objects such as the data available via HTTP, FTP and gopher protocols. Web browsers can then use the local Squid cache server as a proxy HTTP server, reducing access time as well as bandwidth consumption. Squid keeps these objects in RAM or on local disk. Squid servers can be installed in hierarchies to allow central servers to build large caches of data available for servers lower in the hierarchy.

Squid has been in use for some time around SAS EMEA and is performing very well; the software is extremely stable and is delivering seamless access to the Internet for connected clients.

The original proxy servers were installed on HP workstations running release 10.20 of HPUX and Squid version 2.1. This was run on a mix of hardware but typically HP9000/720 workstations with 64MB of memory and about 4GB of disk. This configuration is difficult to support; the hardware has reached an age where failures are becoming common and the increased use of the Internet coupled with growth in the offices has left the configuration under-powered. Our main problem of late has been disk space management; the increased access patterns have left our existing log areas looking undersized at 100MB and our actual cache directories are looking rather small at 2GB.

As a result, we began researching some alternatives in order to maintain the service. Since we were happy with the Squid software itself, and we already had a good understanding of the configurations, we decided to continue using Squid but to review the hardware base.

Since Squid is an open-source project and well supported under Linux, it seemed a good idea to explore the possibility of using a Linux-based solution using a standard SAS EMEA Intel PC. This configuration is a Dell desktop PC with 256MB of RAM, 500MHz Intel Pentium and internal 20GB IDE disk. As Dell has a relationship with Red Hat, it made sense to their distribution. Also, SAS has recently released versions of the SAS product in partnership with Red Hat.

Architecture

The original architecture in SAS EMEA used three central parent Squid caches with direct access to the Internet and child Squid servers in many of the country offices. Some of the smaller countries' operations still connect to the central headquarter caches, and we felt that using less expensive hardware would give us the opportunity to install proxies in these offices. Further, in many of the country operations the SAS presence is split among several offices connected via WAN links; again the less expensive hardware gives us the opportunity to install proxies in these offices. These deployments should improve the response times for web traffic and hopefully reduce the overhead on our WAN links.

Finally, we had some reservations about the resilience and availability of the original infrastructure, and we felt that with revised client and server configurations we could improve the service level of our internal customers.

Our new architecture is not much altered in principle; we still have three central servers, but they now run Linux. We are deploying more child proxies, and we require a three-level hierarchy in some offices. For example, some countries have satellite offices that only connect to the SAS Intranet via a single WAN link to the country headquarters; in these cases we will install proxies at the satellite office with a preferred parent cache in the country headquarters rather than European headquarters.

A new addition to our architecture has been the Trend Interscan Virus Wall product for HTTP virus-scanning. We have installed three virus scanning systems also running Red Hat Linux; these systems are positioned behind the current Squid parent caches, providing a virus-scanning layer between the Squid cache hierarchy and the external Internet. Since the virus scanners are simply pass-through in nature, we simply configure our top-level Squid servers to “round-robin” between them.

Installation

The original HP-UX servers were installed by duplicating a disk image from a known configuration. This was a totally unsatisfactory method for several reasons, not the least of which was that it was difficult to make provisions for maintenance of this image for patches or version updates for Squid, etc.

Our goal was a scripted and automated installation that could be performed quickly by local office staff. We have been pleased with the implementation of this concept, and it has some useful benefits with regard to recovery and configuration management (see below).

We produced a KickStart configuration to build machines for us. KickStart is a tool from Red Hat to automate system installations. Basically we can tell the install how to partition a disk, which packages (RPMs) to install and include some local configuration steps via shell commands. Our KickStart configuration is placed on a floppy disk along with normal Linux boot utilities, and we instruct the KickStart to perform installation from a CD.

This means that for a new proxy server we can arrange shipment of a PC that looks similar to our expected hardware configuration and ship a CD and floppy disk for the remote office to complete the configuration.

The installation process has been automated with three exceptions: users will be prompted for the hostname, IP information and the keyboard type (some of our offices use different keyboards for the local language). The KickStart hard-codes all other choices; for example the installation language is always English, the choice of packages are always the same and the disk always partitions in the same way.

The basic installation from placing the disk in the drive until the reboot with a freshly installed OS takes under ten minutes. This is much quicker than we could do and a huge decrease in the time it would take to perform a HP-UX installation. This obviously has some implications for our backup and recovery procedures (see below).

Configuration Maintenance

We have the usual problems with running our systems: configuration files need updating, software needs upgrading, log files need rotating and processes need monitoring. In the past a seriously inconsistent set of shell scripts and cron jobs had been used, or more normally, configurations had been allowed to diverge. We had replaced some of these with **rdist** jobs but this was not wholly adequate, so we looked around for another tool.

The best tool seemed to be **cfengine**. This allows us to build a common definition of tools and configuration at a central location and then distribute it to all our servers. Implementing cfengine has been highly successful, although it does require some careful planning of the configuration structure and a fairly careful reading of the documentation.

Some of the files we distribute are completely standard, and we can simply have cfengine send them “as is” to the target system. However, some are based on a common template and need alterations for each system. A good example of this would be the main Squid configuration file. In this case we ship a template via cfengine and a small script that knows how to make the transformation. We use the feature in cfengine that allows commands to be run after the receipt of a file so this shell script is run and then Squid is signalled to re-read the configuration file. This way we can keep a centrally controlled and coordinated configuration and know with certainty the status of a remote system.

In turn, our cfengine configuration is built on a local NIS map we maintain; this NIS map simply indexes host names against capabilities. For example, the keyword SQUID-CHILD is used to flag that a machine is a second-level proxy as opposed to SQUID-MASTER. This NIS map is processed to produce classes for use in cfengine; the end result of this is that configuration information is stored centrally, not on each server.

More problematic for us has been maintenance of the installed software. We are running a system largely built around Red Hat 6.2 but since installing some of the proxies, Red Hat released updates that we required. Typically security issues are a priority. We also have some locally derived RPMs, and use a later version of Squid with some options Red Hat does not include, and we use a locally built version of the gated routing dæmon that includes support for some additional routing protocols. Finally, we have some RPMs that were never in the Red Hat distribution.

The obvious step to take was to build an FTP server for updates. We have used the built-in FTP server on a Network Appliance Filer that also contains our distributions from Red Hat. We have an FTP mirror job that pulls the latest updates from a Red Hat mirror site. Our FTP server also has a tree for our local RRMs.

It's taken some time to get the process of updating RPMs correct and we're still not totally happy. The best tool we have currently is **autorpm**. This is configured to look at our FTP server and automatically install Red Hat updates, ignoring those RPMs we didn't install to start with, and to install or update all RPMs inside our local tree.

Our problem here is that autorpm cannot deal with some circular dependencies contained in some RPMs. It's easy to manually resolve this, but we would prefer to automate this process. This seems less the fault of autorpm and more a problem with the actual RPMs.

Back to Installation

This problem with the RPMs has also had an impact on our installation procedures. We were very happy with the installation time being under ten minutes, and it only took a few minutes to apply our old rdist updates for files. Now though, it was taking over ten minutes to apply the RPM updates on some sites and it was a manual process requiring logging in to the system to complete. This was pretty critical for the Squid software where our Squid configuration file requires the newer version of Squid.

We also had an issue with the install itself now; there was no way that an install could complete locally without this intervention from us if we had updated some of the software. This was a fairly important issue for us. There are occasions when some type of failure is experienced at a remote office and, we are not available. In these situations we would like to remotely access the office to simply reinstall their proxy server, possibly on new hardware. This only works if we do not need to intervene in the process.

As a result, we went back and reconsidered one of the initial assumptions—we'd said that we would ship only standard Red Hat CDs and use KickStart, cfengine and autorpm to customize them. We now decided to produce our own Linux distribution largely based on the Red Hat distribution, but including our new and updated RPMs and some configuration files. The idea now was that the initial install would produce a working proxy and then our scheduled, automated jobs would come along later and tidy up any small problems.

Producing our own distribution has been pretty successful; we can produce our distribution from a new Red Hat release in about one to two days. We take the basic Red Hat distribution and remove many of the RPMs from the tree. This is not a very scientific process: we do not remove every RPM, only the ones taking up the most space. We then add our own RPMs to the tree, modify the various control structures in the tree and cut a CD.

Since our new media contains cfengine and autorpm, we can configure the post-install steps of the KickStart process to run these processes on the first boot after install. This should bring our new machine quickly up to date with our current configuration.

However, when we moved to Red Hat 6.2 we hit an interesting issue with KickStart: the new version does not necessarily prompt the user for IP addresses and other network information when they are not present in the KickStart file during an install from CD-ROM. A careful reading of the slightly updated documentation suggests this was deliberate on the part of Red Hat, but was a major headache for us. Ultimately, we rewrote the section of code in the Anaconda installer to restore the original behavior.

Backup and Recovery

We looked at the various backup and recovery options and decided to use the simplest backup procedure available; namely we do not back up the machines. When we looked at the proxies it was apparent that only the log data and the cache structures were unique to any machine. The log data is periodically copied to a central location for analysis and the cache, while valuable, can be cleared and rebuilt over time.

The Linux proxies we have deployed so far have had no major service interruptions or hardware problems, but if we do have problems, we propose to reinstall the remote system. In the case of hardware failure, we expect most remote offices will be able to find a spare PC and repeat the installation.

As a result, not only are we not taking backups, but we do not need to make any provision for resilient hardware or technologies for mirroring, etc. In fact, using specialized hardware would reduce our availability and resilience since we would not have spare hardware at the remote office to make a replacement, whereas we have many standard desktop PCs of a sufficiently similar configuration that our KickStart procedure will work on them.

We expect that in the event of a failure the remote office can recover the service in around 20 minutes, given that a PC is available. We have made some effort with the client browser configuration to make this transparent to the desktop user. This gives us a highly available solution.

Client Considerations

The old proxy configuration relied on the clients being configured to use a named proxy server explicitly. For the sites where more than one proxy server was available, the hostname used for the proxy server was in a domain managed by **lbnamed**. The version in use had been modified slightly but was still unsuitable for our use. lbnamed used **rpc.rstatd** to get loading information from machines in a pool, and then depending on the weighting, lbnamed will return different hosts, thus creating some limited load balancing. In practice this does not distribute effectively although it has the useful feature that if a machine is unreachable it will be removed from the pool. Unfortunately, this useful feature is undermined by the fact that only load is used (in the basic Perl version) to weight hosts. If our Squid server dæmon dies the load on the machine tends to to be reduced, which can leave a machine where a failure has occurred at the top of the pool. There is an implementation in C that can look at other factors apart from load but some experimentation with this was not fruitful. Our overall impression was that, as previously installed, this was more successful than a standard DNS round-robin would have been.

Our testing was performed against our three new Linux proxy servers, and one factor we noted was that response time could be improved if we sent the HTTP query to the machine most likely to have the object in cache; of course this seems a fairly obvious but not immediately useful observation.

In fact, the idea of intelligently selecting the cache to query is a fairly simple thing to achieve using the Proxy Auto Configuration (PAC) file feature supported by most mainstream browsers. This basically entails having a web server somewhere that can return a proxy PAC script.

Our task was made doubly easy because someone had already produced some sample code for PAC files that balanced queries across several servers. We took as our base the work done at Sharp on a “Super Proxy Script” using URL hashing. This is a simple but ingenious idea that hashes the URL being requested and then returns a proxy to use. This is statistically random in terms of load distribution over large numbers of URLs, but repeated queries for the same URL will always return the same proxy. We also make use of the ability in the PAC script to return an array of proxy servers; the effect is that if the first named proxy fails then queries by the client are routed seamlessly to the next proxy in the list.

At the headquarter sites, we return arrays based on two or three proxy servers depending on the campus location. For sites where only a single proxy is available, we do not use URL hashing and only return a pair of proxies, namely the local proxy server and a central server for fallback.

The use of this central fallback for remote users is the feature that gives us the most resilience. Should a remote proxy fail the remote clients for that proxy will detect the failure and use the central host, the clients will check every 30 minutes (MSIE and Netscape) to determine if the original server is active and return to using it if it is.

In fact, since we have some clients using Microsoft Internet Explorer version 5.0, we name the proxy.pac script as wpad.dat. This allows “unconfigured” IE5 clients to locate the wpad.dat file automatically using the WPAD method of searching for a URL of the form http://wpad.local.domain/wpad.dat.The use of WPAD is not particularly critical to us but it is a useful feature. Reviewing the logs during implementation suggests that we may be saving our help desk some calls from mobile users who would otherwise have required help setting their proxy manually when visiting other offices.

Currently we use Apache web servers to return the PAC scripts, and the Apache runs either on the local proxy host or another locally available web server. It is possible that using one of the stripped-down web servers, for example, several are implemented in Perl, may be more secure and represent less overhead. We have not explored this approach yet.

Our WPAD servers currently have multiple address records in DNS so in the event of a single WPAD server failing, we have some resilience. For sites with only a single WPAD server, we rely on new client sessions using previously cached settings.

There is a small flaw in this approach where remote sites have multiple proxies: the URL hashing carefully selects which local proxy to use but then the proxy simply round-robins the query over the three central systems. We could use some of the facility of Squid to exchange cache digests so the local proxy would forward requests to the central server most likely to generate a hit, but in practice the cost in bandwidth on the WAN links makes this ineffective. Instead we let the round-robin query any central server and then have the central servers use cache digest exchange to generate a hit if possible.

Upgrades

We anticipate that we will want to keep our proxy servers up to date with a reasonably current Red Hat version. While we will rely on cfengine and autorpm to make the small alterations in configuration, we do not expect that upgrading the entire OS over the network is really feasible for us.

So, instead, we intend to ship our fresh installation to the remote office and have them carry out a new install if we want to upgrade. We expect this will occur about three or four times a year possibly. Because we have such a clean installation process and tight control over configurations, there's little penalty for making such frequent upgrades. Since the remote office would retain the previous media, regression to an earlier version should also be fairly straightforward.

We are very eager to take these frequent updates as we experience many problems on the HP-UX proxies directly related to lack of software and patch updates. For example, we see HP-UX problems occur for which patches are available, and have an old version of Squid and many tools we try to run since old versions of Perl are available.

An upgrade can be done during a normal business day for a remote office, and for the ten or 20 minutes downtime the clients will simply fall back to the central standby servers, or for sites with multiple servers, they use their own local fall back servers.

Monitoring

For planning purposes the most useful data is a historical view of the activity the cache is seeing. We gather this data using MRTG (Multi Router Traffic Grapher) and the built-in SNMP agent in Squid. This has been a little awkward to configure, but we are able to create some useful graphs of the performance of the proxies. We collect many metrics from the proxies that are available from an internal web page. We have some code that generates an index page of all servers by walking our NIS map of hosts (see above); this code also includes a thumbnail of some key metrics. We are particularly interested to see the trends for cache hit and cache miss times as well to track any overall growth in requests.

We also use the Calamaris tool (see Resources) to produce snapshots of the status of the proxies and some analysis of the logs.

We also have a copy of the HP OpenView network management products. Currently we are using this to monitor the status of the machines only, but we plan to customize it to monitor the health of the remote Squid software to simply alert us if the software fails.

We have also started to use cfengine running from cron every five minutes to check the health of various processes and to attempt an automatic restart.

Additionally, we deliver Internet usage reports to the local offices. Currently we collate the logs into an SAS dataset and use SAS reporting tools to produce reports, but we also have products like SAS IT Service Vision and SAS WebHound that are able to produce similar analysis of traffic. These are powerful tools, and we can use them to provide a much fuller analysis of the data.

We have been pleased with the stability and performance of our Linux solution. There's no doubt that the reduced hardware costs are allowing us to install proxies in locations where previously it was not cost effective to build more resilience for other locations. Since our new configuration is more resilient and, by and large, better configured than the previous one, we have less problems than with the old proxies.

The worst problem we have seen operationally so far has been file system corruption. We have had a remote proxy suffer a power failure and then fail to boot because of file system problems. As an interim measure we have amended the startup code to be more tolerant of these failures, but a more long-term solution may be to use one of the log-based file systems that are becoming available.

We are beginning to see that more local links to ISPs are being deployed in our offices. As a result we will need to fine-tune our configuration to support HTTP virus scanning at an office level and direct access from the proxy to the Internet. In practice this will simply mean adding some tasks to cfengine and autorpm to install and configure the new modules.

Since we now have a procedure to produce our own Linux distribution media, it is likely we will review a more generic Linux deployment internally. This will occur to some extent anyway now that SAS products are available on Linux; we hope producing standard distributions for internal use will give us some loose configuration control. We will likely consider what other functions might usefully run on Linux; for example, we may move some DNS/BIND functions to Linux.

Resources

Ian Spare is a consultant currently working for the SAS Institute in Germany. He prefers to spend his time snowboarding or on skis but, when not on the snow, manages to support and manage a mix of UNIX and Linux systems around Europe, the Middle East and Africa. He has no children but does have three dependent cats to support.

Alternatives for Dynamic Web Development Projects

Dennis Gesker

Issue #83, March 2001

Dennis provides a starting point for developers seeking solutions for their web application development requirements.

I was recently afforded the opportunity to work with the internet department of a large medical publishing firm located in Philadelphia, Pennsylvania. The department assisted the company's various marketing departments with regard to content destined to be posted on the Internet as well as coordination of scheduling and placement of content on the company's web site. Ultimately the department served as a conduit between various marketing and product management departments at sites in Philadelphia, New York, Chicago and the Baltimore-based IT department that physically housed the company's web servers.

As a result of these activities, there were a great many files finding their way into the department via a number of paths. Most arrived by e-mail, some on diskette via interoffice mail and some were uploaded directly to ftp servers in Baltimore. This created a flurry of activity as the marketing staff and developers in the department attempted to organize and coordinate a variety of content. To add to the confusion, there were also numerous versions of documents that were submitted due to last-minute changes made by the supported departments. My task was to develop an interim solution to reduce the confusion created by content submission activity by organizing all the various content in one central application. I attempted to do this by building a small web application prototype that would serve to track each project's title, manager, department, department code and cost accounting code.

The application needed to be built quickly and cheaply because constraints in the purchasing and tenuous acquisition processes ruled out any new requisitions until the beginning of the next fiscal quarter. It needed to run on standard PC equipment that was readily available in the department. Technical support for the application by experienced system administrators would be limited. The applications needed to be light, fast, easy to use, stable and easily maintained by the department's staff that approached web development from a marketing, as opposed to technical, perspective. Despite the fact that this was the Department of the Internet, the focus of the department's activities was on content coordination. The IT department administered the heavier-duty web servers and telecommunications that were off-site.

Objective

This document will relay the lessons learned from my investigation into the software tools required and available for building a rudimentary, dynamic internet application. Consideration will be given to commercial and free software alike. We'll begin with some definitions and background of the elements of a dynamic web application and then continue on to a tour of some of the available technologies. It is my intention that this article serve as an informative starting point for developers and would-be developers as they begin to seek solutions for their web application development requirements.

What Is a Dynamic Web Site?

Advanced web designers often use a scripting language called JavaScript and a system of naming the parts of the web page—the Document Object Model, or DOM—together with HTML and CSS to create dynamic content on a page. These effects are sometimes called Dynamic HTML or DHTML. I, however, am limiting the scope of this project to building a web site that will generate dynamic content by interacting with a database. I was not interested so much in effects displayed to the user but in focusing on building an intuitive web application. The project scripts will be executed at the server, not at the client as occurs with JavaScript. The tools and methods presented here will not limit the addition of interactive or stylistic features to the application at a later time.

What Is a Database?

Data is defined as a general term meaning the facts, numbers, letters and symbols processed by a computer or communications system to produce information. In a computer system these items are typically stored in files. A collection of related files is a database. Within these files, records of items are organized into rows and columns. In this case, I need something more sophisticated than a simple collection of files.

The project requires a RDBMS, or Relational Database Management System. An RDBMS is a software package that stores data in rows and columns as tables. Various tables can be related to one another in order to answer questions posed by the end user. These questions are known as queries. The RDBMS is a concept first introduced by Codd in an academic paper in 1970 but was not commercially available until the mid 1970s. The RDBMS responds to all queries whether they ask the RDBMS to retrieve, add, update or delete data from the tables.

Sometimes the RDBMS that services a web site is called the back end. The web pages that the end user will see is often called the front end. Questions are posed to the back end using a language called SQL (Structured Query Language). In short, if there is data that needs to be processed, a DBMS of some kind is required. The most popular at this time are Relational DBMS, that respond to questions/commands via SQL.

Choosing a Database

Oracle

One product that has become synonymous with the word database is Oracle. Oracle (http://www.oracle.com/) is largely responsible for the current popularity of the relational database. Over the years, its database server has become justifiably respected for being full of features, fast and reliable. Oracle is also supported on Linux, and it appears that Oracle is committed to Linux as a platform.

However, there were two primary reasons not to use Oracle for this project. First, the hardware requirements were well beyond the capabilities of the machine used to develop and serve this application. As a rule of thumb, 800MB of disk space is needed with 256MB of memory. Second, Oracle would have been too expensive an alternative for an application of this size and temporary nature. Even at the minimal licensing fee of five named users in perpetuity for a single server the cost of using this software would have been ($160/user) $800.

MySQL

There are several open-source database options available to developers. In conjunction with web sites, MySQL (http://www.mysql.org/) appears to be a very popular choice in the Linux community. MySQL is a very fast, multithreaded, multiuser and robust SQL database server. MySQL is now also open source and has recently formed a strategic alliance with VA Linux Systems, a company that sells and supports Linux-based computer systems. MySQL was first released to the public in November 1996 and has always been available with source code. MySQL has proven to be a lightning fast and reliable database solution for a growing number of companies such as SGI, ValueClick, Nortel/Insight, Tucows.com, Cisco and many more.

So it would seem that MySQL was more than up to the task of the humble application described at the beginning of this document. It is considered very fast with large record sets, and the MySQL Manual reports production systems with upwards of 50,000,000 records. Further, there seems to be a growing relationship between the team that develops MySQL and the team that develops PHP. The increased popularity of this dynamic duo, coupled with boundless enthusiasm from core developers of both technologies, culminated in a meeting of the minds in Israel earlier this year. This resulted in the MySQL library being packaged with the PHP 4.0 distribution, in addition to an agreement to help each other improve the performance quality of product integration whenever the opportunity arises.

However, MySQL does have some shortcomings. One of these shortcomings is in the area of transactions. Tim Kientzle eloquently and succinctly discusses transactions in his July article written for *Dr. Dobb's Journal*:

> A transaction is a set of related changes to a database. The SQL standard specifies that an entire group of updates can be issued to the database and then either committed or rolled back as a unit. This lets you, for example, transfer money between accounts stored in different database tables by adding the money to one account and then trying to subtract it from another; if the second update fails, you can undo all of the changes at once.

While lack of transaction support will not immediately be an issue in our application, the threat of “feature creep”, even in an application of this nature, compelled me to seek an alternative system that does support transactions. Further, MySQL has only limited support for foreign keys. The foreign key is an important concept of the relational model. It is the way relationships are represented and can be thought of as the glue that holds a set of tables together to form a relational database. Another area where I found MySQL to be wanting was in subqueries that it does not support. A subquery occurs when a developer nests one SQL statement within another SQL statement. Again, while my application is small and likely would not run into too much difficulty with MySQL's support of foreign keys, subqueries and transactions, it was still another needling in the direction of an alternate RDBM.

PostgreSQL

Some of the concerns that I had with MySQL seemed to be addressed by the team developing PostgreSQL (http://www.postgresql.org/). The version of PostgreSQL that ships with Red Hat 6.2 is version 6.5.3. This version already had support for transactions and subqueries, but it did *not* have extensive support foreign keys. However, the PostgreSQL team had recently released version 7.0.2 of the database that does support subqueries.

The PostgreSQL FAQ reports that PostgreSQL has most of the features present in large commercial DBMSs, like transactions, subselects, triggers, views and sophisticated locking. It has some features that other databases do not have, like user-defined types, inheritance, rules and multiversion concurrency control to reduce lock contention. But this functionality seems to be at the expense of the speed afforded MySQL. In comparison to MySQL or leaner database systems, PostgreSQL is slower on inserts and updates because it has transaction overhead.

Similarly to MySQL and in great contrast to Oracle, the minimum system requirements of PostgreSQL are light. The PostgreSQL administrator's guide reports that although the minimum required memory for running PostgreSQL can be as little as 8MB, there are noticeable speed improvements when expanding memory up to 96MB or beyond. The rule is you can never have too much memory.

Check that you have sufficient disk space. You will need about 30MB for the source tree during compilation and about 5MB for the installation directory. An empty database takes about 1MB, otherwise it takes about five times the amount of space that a flat text file with the same data would take. If you run the regression tests you will temporarily need an extra 20MB.

The machine I used easily meets these system requirements. I don't expect the drop in speed to be an issue. I'm satisfied that PostgreSQL addresses my concerns regarding MySQL and Oracle. I decided to use PostgreSQL for this project.

Choosing a Scripting Language

As with database software there are many scripting languages available for the Linux platform. Several are appropriate for use as a server-side scripting language in a dynamic web application. The purpose of the server-side scripting language is to tie together the user interface presented to the user, which here will be written in HTML and accessed via a web browser, and the back end of the application or server system and database used by the application.

ColdFusion

One commercial product considered as an option was ColdFusion from Allaire (www.allaire.com). The Allaire product is widely used, and the server portion of the environment is available and fully supported for Linux platforms as well as on Windows NT. Allaire also offers a very nice development environment as part of this package called ColdFusion Studio that requires either the Windows 9x or NT platform to run. Both components are available via download for a period of evaluation or purchase from the Allaire web site.

The components of the ColdFusion package were built to support the ColdFusion Markup Language (CFML). CFML has a syntax that is similar to HTML in that there are opening and closing tags that provide functionality beyond normal HTML. These specialized tags intermingle or are embedded in the HTML application or page. The ColdFusion server works with the web server to intercept these special tags to allow for interaction with the database and server providing the back end of the application.

The impressions that I carry of this product and environment are very positive. It seems to be a very capable platform that is easy to set up and use. There also appears to be a large group of web-site developers using this program. I believe that casual interactions with colleagues are often very helpful when becoming familiar with a new language or software product. The similarities of the CFML to HTML would also make it easy for a novice to be up and running quickly. Additionally, ColdFusion works with Apache, another check in its plus column.

Drawbacks are, however, that the system requirement for the Linux version of the ColdFusion server, 512MB of RAM, is beyond the capabilities of the system that I have available for this application. The July 17, 2000 price list posted on the Allaire site reports a price tag of $1295 US for the ColdFusion Server 4.5 Professional for Linux. The Windows-based development environment will run another $495.00. If the budget for your application can absorb the licensing cost of ColdFusion, it deserves consideration.

Perl

In the Linux community Perl carries about as much respect as any programming language deserves to expect. It seems that beyond its use as a web-scripting language there is little that Perl cannot handle. Written by Larry Wall, open-source Perl was first released in 1987. Perl's latest version released this past March is version 5.6 and is available from http://www.perl.org/.

Perl is a very mature and viable alternative as a web-scripting language. There is a large developer community and a great deal of support available. It can be tightly integrated with the Apache web server and is available with most, if not all, Linux distributions. However, what I was hoping to find was a simpler solution for attaching my web pages to a database back end. Just because Perl is the most obvious solution didn't necessarily mean that is was the right solution. The simplicity criteria weighed heavily in my decision to examine ColdFusion and partially ruled out Perl in this case. I wanted to get my application up and running quickly with as small a learning curve as possible.

PHP

PHP (http://www.php.net/) is an open-source, HTML-embedded scripting language. Unlike Perl, which was born as a tool to assist in system administration, PHP was designed from the ground up to work with web pages. It seems to borrow from many programming languages that have preceded PHP, including Perl and C. The PHP FAQ discusses the differences between PHP and Perl:

> The biggest advantage of PHP over Perl is that PHP was designed for scripting for the Web, where Perl was designed to do a lot more, and because of this, can get very complicated. The flexibility/complexity of Perl makes it easier to write code that another author/coder has a hard time reading. PHP has a less confusing and stricter format without losing flexibility. PHP is easier to integrate into existing HTML than Perl. PHP has pretty much all the "good" functionality of Perl: constructs, syntax and so on, without making it as complicated as Perl can be. Perl is a very tried and true language; it has been around since the late eighties. But PHP is maturing very quickly (www.php.net/FAQ.php#9.4).

From this discussion it would seem that PHP would be a good fit for the application I was building. There has also been some discussion as to how PHP compares with ColdFusion. The PHP FAQ gives a good summary but also points to a comparison of the two packages laid out by Mike Sheldon. Mike's comparison (http://marc.theaimsgroup.com/) appears even handed. He points out some strengths and functionality of ColdFusion that I had not considered. One of the most interesting is an abstraction layer to database interaction. In ColdFusion, the developer uses the ColdFusion administrator to set up a data source. Whether the source is Oracle or an ODBC connection is inconsequential with regards to the CMFL tag used to interact with the source. In PHP, each database that we may want to use has a set of functions that are specific to the chosen database. Mike also points out that ColdFusion is bundled with a search engine called Verity. PHP has no bundled search engine and does not have an integrated development environment.

Two areas I was surprised to hear that PHP had weaknesses in were that of error handling and the ability to handle dates. Because of the simple nature of this application, I don't see either of these weaknesses becoming a problem. I only intend to use one database, and there is a function set for each database under consideration for this application. I don't see the data abstraction offered by ColdFusion as being a bonus. In the final analysis, I selected four primary components for this application: GNU/Linux operating system, PostgreSQL RDBMS, PHP server-side web-scripting language facilities and the Apache web server.

Conclusion

There are a myriad of software options available for the Linux operating system. The alternatives all have strengths and weakness that arise from the software's efforts to serve a particular primary need. I would recommend that those interested in deploying one of these software packages use a reasoned approach in determining which package best suits their needs. The value of developers and consultants is only partially derived from their mastery of a particular technology. The understanding of the particular processes by which they hope to influence their deployment of an information-technology solution should be of primary concern. Because there are many high-quality alternatives available to address a given information-technology challenge, it is the technologist's ability to grasp and deal with the key issues of the project in question that will ultimately determine of its success or failure.

For my project, the combination of Linux, Apache, PostgreSQL and PHP was appropriate. This combination will likely not be appropriate for all projects. However, I hope that you find this article an informative first step in your investigation of alternatives for your next project.

Resources

Dennis Roman Gesker (drgst30@katz.pitt.edu) is currently a candidate for the degrees of MBA and MS-MIS in the dual-degree program at the Katz Graduate School of Business, University of Pittsburgh. Upon graduation he will be joining the management team of Alamon Telco, Inc. in Kalispell, Montana.

As the Log Scrolls By…

Gaelyne R. Gasson

Issue #83, March 2001

Gasson shows how a few tweaks to Apache's httpd.conf file can provide a colorful web log file.

As a web hosting company, there are times when it's vitally important to see what our Apache web server is dishing up to the outside world at any one time —and to see this as quickly as possible.

Just as a system administrator needs to be able to monitor system log files, a web administrator should be able to do the same with web logs. Noting the number of utilities that display system log information in real time, I was sure there'd be similar programs for monitoring web logs. After a search through Freshmeat.net and other on-line resources, I didn't find anything that met all of my needs. Several came close, but most would only monitor one file, and the few that monitored several files would leave me lost trying to wade through tabs for the 30 or so logs that I watch.

The solution I found isn't in one program, but in making a few changes in Apache's httpd.conf file I found I could have a specialized disposable log file containing only the information I require, for all of our web hosts. The “disposable” monitoring log is then displayed using colortail (with additional configuration settings) on an external monitor in our workshop. I can see at a glance which of our hosts have current web activity, where the traffic is coming from and the pages that are being accessed. This has also allowed us to deal quickly with problems such as script kiddies and rogue search engine robots. The system has worked so well for us that we added system logging to it as well.

httpd.conf Changes

In addition to the LogFormat for general logging, I added a new format labeled “webmonitor”:

```
LogFormat "[%v] %h %u \"%r\"%>s%b\n\"%{Referrer}i\" \"%{User-Agent}i\"%t" webmonitor
```

This displays log information with the Referrer and User-Agent on a second line, making it clearer to read. The log file could be in any format—even the "common" one we use for standard logging. I decided to change it for purposes of clarity and because Apache is flexible enough to allow this.

Since graphic files such as GIFs, JPEGs or PNGs files can clutter up the display, I exclude them by adding the following three lines to the general log section in httpd.conf:

```
SetEnvIf Request_URI \.gif$ unwanted
SetEnvIf Request_URI \.jpg$ unwanted
SetEnvIf Request_URI \.png$ unwanted
```

We use name-based virtual hosts, and each host has their own <VirtualHost></VirtualHost> container. In addition to their permanent log file, we add an additional CustomLog command for our webmonitor file for each of our hosts. For example:

```
<VirtualHost someisp.com>
...
CustomLog /var/log/httpd/someisp.com-access_log combined
CustomLog /var/log/httpd/webmonitor_log webmonitor env=!unwanted
...
</VirtualHost>
```

Our addition is:

```
CustomLog /var/log/httpd/webmonitor_log webmonitor env=!unwanted
```

/var/log/httpd/webmonitor_log is the path and filename for our monitoring log file, and Apache will create it for us at startup if it doesn't already exist. webmonitor is the name of our custom format log defined in the LogFormat section above. **env=!unwanted** sets Apache so it doesn't log any items we've listed in the SetEnvIf lines (the .gif, .jpg and .png file extensions). This way we don't see graphic file requests but we do see all others.

Adding System Log Information

The ability to see what's happening on the server with an external monitor proved so useful that we also included system logging information in the same file. To do this, we edited /etc/tem syslog.conf to include the following command:

```
kern.*;authpriv.*;*.crit;*.error;*.warning;*.emerg /var/log/httpd/webmonitor_log
```

Colortail

Colortail was written by Joakim Andersson (pt98jan@student.hk-r.se) and is available from www.student.hk-r.se/~pt98jan/colortail.html under the GNU Public License.

While we could simply tail the webmonitor log file, adding color to the display is a nice touch and gives us an indication of which web host is seeing activity even if we happen to be on the other side of the workshop.

Colortail comes with several sample configuration files; none really suited web logs, although conf.xferlog comes close. After a bit of tweaking, this is the format we've been using. It's a hybrid as it includes both web and system-log-related items.

Listing 1. colortail.conf

Displaying the Colortail

To use colortail locally, you could use a command such as:

```
colortail -f -k /etc/colortail /var/log/httpd/
 <@cont_arrow><\#229><@$p>webmonitor_log &
```

This is fine except that it doesn't allow us to have it on screen all the time, and I'd often need to switch to the particular console or X window displaying the log.

To be able to monitor activity better, we display the colortail output on a Commodore 128D computer connected to the system. Our particular set up has our C128 connected to a private server using a null modem and PPP connection. From here, we log in to the server with the log files. You can use any inexpensive spare computer you may have lying around for this purpose, as long as it's capable of handling ANSI or VT100 emulation and has an 80-column display. PPP isn't a requirement.

Rather than type the command to start the colortail on the Commodore machine, we use a nightly cron program that rotates the log file and then sends the colortail output to the PTY device. See Listing 2 for the file used for this purpose.

Listing 2. Cron Program

Wrap Up

There are probably as many ways to monitor log files as there are Linux users, but that's part of the fun. While there really isn't anything “new” about using colortail to display log files, this is a different combination of resources from those I've read about, and it works for my requirements. Hopefully, this article will help others looking for a way to view real-time web activity.

Gaelyne R. Gasson (gaelyne@videocam.net.au) is a web administrator in South Australia. Using the web monitoring methods described above, she can tell at a glance if someone's watching her webcam (http://gaelyne.com/webcam/).

Advanced search

Using xinetd

Jose Nazario

Issue #83, March 2001

Jose demonstrates how to start configuring and tweaking xinetd.

Replacing **inetd**, **xinetd** provides access control, improved logging and resource management. It has become the standard Internet super dæmon for Red Hat 7 and Mandrake 7.2. This article is designed to get you started with using some of its features—hopefully some of its more interesting ones—and is based on features available in xinetd 2.1.8.8pre3.

Preamble

The original author of xinetd, Panagoitis Tsirigotis (panos@cs.colorado.edu), seems to have dropped the project. Rob Braun (bbraun@synack.net) has picked up the project and is now responsible for maintaining the package. One problem I noticed with the package in its current state was that I had to add a couple of header files to get select( ) to work on my old libc5 system. Should you need them, they are as follows:

```
xinetd/internals.c.orig
Fri Jun 16 19:00:15 2000
+++ xinetd/internals.c
Fri Jun 16 19:00:53 2000
@@ -12,6 +12,8 @@
 #include <time.h>
 #include <fcntl.h>
 #include <syslog.h>
 #include <unistd.h>
 #include <sys/time.h>
 #include "sio.h"
```

About xinetd

xinetd replaces the common inetd lines with bracketed, expanded syntax. In addition, new possibilities are given for logging and access control. While inetd allows control for TCP connections using Venema's tcp_wrappers software (tcpd), you cannot control UDP connections. Also, it doesn't do well with RPC (portmapper) type services. Additionally, while you can control the rate of connections using inetd (by appending a number to the wait or no wait argument, for example, nowait.1 for one instance per second), you cannot control the maximum number of instances. This could lead to process table attacks, for example, an effective denial of service. By using xinetd, we can thwart this.

I usually start xinetd with the following command, placed in my startup scripts where Internet services are started:

```
/usr/sbin/xinetd -filelog /var/adm/xinetd.log -f /etc/xinetd.conf
```

This tells xinetd to log everything to the file /var/adm/xinetd.log and use the configuration file /etc/xinetd.conf. The bulk of this article will deal with this configuration file.

Compile-Time Options

The three compile-time options you should pay attention to that provide added access control are libwrap, loadavg (a threshold monitor for load averaging) and IPv6 support. As with most libwrap-aware dæmons (like **portmapper** and **sendmail**), the option “with-libwrap” in the configure script tells xinetd to be built linking in support for the tcp_wrappers file /etc/hosts.allow and /etc/hosts.deny. These options for xinetd work exactly as they do for inetd and support all of the xinetd-controlled dæmons. Note that if you're starting from scratch with xinetd, using tcpd is no longer needed as access control is done within xinetd. However, this support for libwrap is useful if you're migrating from inetd/tcpd and don't want to change your access files too.

The second interesting configuration option is support load average monitoring, accomplished using the with-loadavg option in the ./configure script. sendmail supports dropping connections at high load, presuming that it has spun out of control and is taking down the machine. The max_load option can be enabled using this configuration option to limit connections to any or all services based on the load average of the machine.

Lastly, the configuration option to add IPv6 support is accomplished by using the with-inet6 capability in ./configure. This adds xinetd support for IPv6 addresses and connections. Note that your kernel (and network) must support IPv6 for this to be effective. IPv4 support is maintained, of course.

The Configuration File

The xinetd configuration file, usually /etc/xinetd.conf, can be built by hand or automatically from an inetd.conf file. The former is more time consuming and prone to errors; the latter is readily accomplished using the **itox** utility or

xconv.pl script. Although the itox utility is being dropped in favor of the xconv.pl script, it is still useful. However, note that running it repeatedly will overwrite the existing file. Both itox and xconv work in the same way, but we'll show it for itox:

```
$ itox < /etc/inetd.conf > xinetd.conf
```

The newer utility, xconv, understands comments and the use of tcpd better than itox does. For itox you have to specify the directory where dæmons live, such as /usr/sbin. The first section you may want to include is the defaults section. This gives, as the name implies, defaults for the xinetd service:

```
defaults
{
   instances        = 25
   log_type         = FILE /var/adm/servicelog
   log_on_success   = PID HOST EXIT
   flags            = NORETRY
   log_on_failure   = HOST RECORD ATTEMPT
   only_from        = 129.22.0.0
   no_access        = 129.22.210.61
   disabled         = nntp uucp tftp bootps who
                      shell login exec
   disabled        += finger
}
```

Immediately, we can see the syntax of a xinetd configuration parameter: <directive> <operator> <value>. The directives that xinetd understands are listed in Table 1. Directives we'll ignore here are flags, type, env and passenv. We'll talk more below about only_from and no_access, plus logging options.

Table 1. Directives for xinetd

Operators are quite simple, either = or +=. Using =, the values on the right are given to the directive on the left. += is also quite intuitive and is used to append values to an already defined directive. Without it, earlier directives are overwritten. This can also be used to spread access lists or, for example, over multiple lines.

Service descriptions are given by the format:

```
servicename
{
        directive = value
        directive += value
}
```

Servicename must be listed in the /etc/services to occur on the proper socket and with the proper protocol.

A Word about Access Control

Actually, a few words about how xinetd does access control. First of all, xinetd controls connections, not packets. It's a userland dæmon, just as inetd is. As such, while it would break a SYN or connect() attempt from a host that is prohibited from connecting to a service, it will not break “stealth” scans such as a FIN scan [a port scan utilizing TCP packets with the FIN flag set, often performed with a tool such as NMAP]. Don't rely on xinetd to be a firewall to break portscanning. A resourceful intruder will be able to use this information to gather access-control lists for your various services. Luckily, this can be logged by xinetd, and your paranoia sensors should go off when you review your logs.

Secondly, xinetd, as of 2.1.8.8pre3, performs name lookups when a system attempts to connect. Previously, it used to do lookups at startup, but this has been changed.

Using access control is really quite simple. The first directive is only_from, which lists the networks or hosts from which the we will accept connections from. This directive sets up rules that can be overridden by no_access. You can use network numbers, such as 10.0.0.0 or 10, or network names, including *.my.com or .my.com with this directive. Host names and IP addresses of hosts also can be used here. Use the directive 0.0.0.0 to match all hosts and to listen to all addresses. Denials are parsed once the criteria are met by using the no_access directive. Again, networks or hosts can be specified.

Service Configurations

Let's have a look at some basic applications of this information. The first service we'll look at is the **echo** service, which is internal to both inetd and xinetd.

```
service echo
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = root
        type            = INTERNAL
        id              = echo-stream
}
```

Echo runs as root, is a tcp stream and is handled internally. The id directive of echo-stream would show up in the logs. In the absence of only_from or no_access directives, access to this service as configured is unlimited.

Now, let's look at a regular service, in this case the daytime service:

```
service daytime
{
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = nobody
        server          = /usr/sbin/in.date
        instances       = 1
        nice            = 10
        only_from       = 0.0.0.0
}
```

Again, anyone can connect to it, but we specify an executable to run (as nobody) to return the information. This one doesn't extend the previous example by much. We now look at another service for secure shell version 1. This was done to prevent resource exhaustion by **sshd**.

```
service ssh1
{
        socket_type     = stream
        protocol        = tcp
        instances       = 10
        nice            = 10
        wait            = no
        user            = root
        server          = /usr/local/sbin/sshd1
        server_args     = -i
        log_on_failure  += USERID
        only_from       = 192.168.0.0
        no_access       = 192.168.54.0
        no_access       += 192.168.33.0
}
```

Here, we build on what we were doing before. Recall that sshd needs to be started with the **-i** flag when it is started from a super server like inetd or xinetd, so we place that in the server_args directive. Note: adding the flags to the server directive will cause it to fail. Only ten people can use this service at any one time, which is not a problem on the server this example was taken from. We log, in addition to the default information, the user ID of the connecting party as described in RFC 1413 if they are unable to connect. Lastly, we have two networks listed which cannot access this service.

Logging and xinetd

The logging directive understands several values that can be used to get information about your server (see Table 2).

Table 2. Various Logging Directive Values

As such, typical lines to add specifics about logging may look like those listed below. For a service that successfully connects, we usually want to log the process ID of the service spawned, the host that connected and when it exited:

```
log_on_success   = PID HOST EXIT
```

This will give us useful information for debugging and minding normal server operations. For failures, we log what we would expect:

```
  log_on_failure  = HOST RECORD ATTEMPT
```

Here we log the host that connected, the reason the connection was denied and additional information about the connecting host (sometimes the user ID that attempted to connect). These are recommended baselines to give you a good view of your server.

Recall that above, in our defaults section, we were logging to /var/adm/servicelog. We have directed all the information, both failures and successes, to be logged by xinetd. Most of our information will look like this:

```
00/9/13@16:05:07: START: pop3 pid=25679 from=192.168.152.133
00/9/13@16:05:09: EXIT: pop3 status=0 pid=25679
00/10/3@19:28:18: USERID: telnet OTHER :www
```

Using this information, it is easy to debug xinetd and normal operations. It is also easy to spot security issues such as connection attempts that you want to try to block. Simply grep for “FAIL” in the logs, and these entries will stand right out:

```
00/10/4@17:04:58: FAIL: telnet address from=216.237.57.154
00/10/8@22:25:09: FAIL: pop2 address from=202.112.14.184
```

Acting on security issues requires another article, but, suffice it to say, don't take the address reported as solid information, since it can be forged.

The xinetd.log file, by contrast, contains information from xinetd. This can be useful in debugging connections that give errors.

```
00/10/25@21:10:48 xinetd[50]: ERROR: service echo-stream,
accept:
Connection reset by peer
```

Reconfiguring xinetd

You can edit the xinetd.conf file while xinetd is running. To get it to reconfigure, send the signal SIGUSR1 to the xinetd process:

```
# ps -ax | grep xinetd
   50  ?  S    5:47 /usr/sbin/xinetd -filelog /var/adm/xinetd.log -f /etc/xinetd.conf
# kill -SIGUSR1 50
```

Tail the -filelog you are using to make sure that it restarted and adjusted the changes you made. Definitely do this before you log out and make sure you can log back in if this is a remote connection.

Note that using -HUP, as one does for inetd to reconfigure it, will actually cause xinetd to cease operation. This is, by design, to thwart hackers who reconfigure your xinetd and attempt to reload it without understanding the documentation.

When to Use xinetd

Personally, I use xinetd for almost all of my services; the only one that sees a significant performance hit is my web dæmon Apache. Too many processes have to start too fast for it to be time efficient. DNS services should also *not* be loaded into xinetd; the performance hit is too large.

I do, however, run sendmail out of xinetd, allowing fine-grained control over who can connect. My configuration for sendmail looks like this:

```
service smtp
{
        socket_type   = stream
        protocol      = tcp
        wait          = no
        user          = root
        server        = /usr/sbin/sendmail
        server_args   = -bs
        instances     = 20
        nice          = 10
        only_from     += 0.0.0.0
        no_access     += 129.22.122.84 204.0.224.254
}
```

Even on a high-traffic mail server the performance hit is negligible. I have also loaded sshd into xinetd to prevent a process table attack on it.

Conclusions

I hope this article has been helpful to you in getting xinetd configured and tweaked for your needs. As you can see, the features it offers are tremendously more than inetd, even with tcp_wrappers in place. Solar Designer (http://www.openwall.com/) has a patch available for a slightly older version of xinetd, version 2.2.1, that allows for instance control on a per IP basis, which helps stop simple process table attacks. Note, however, that simple forgery can get around this. I do not know if this patch has been applied to later versions of xinetd or if it can be.

José Nazario is a biochemistry graduate student nearing the completion of his PhD. Side projects include Linux and other UNIX variants, software and security-related matters, and hobbies outside of his office like fly-fishing and photography.

Archive Index Issue Table of Contents

Advanced search

Open Source in MPEG

Leonardo Chiariglione

Issue #83, March 2001

Covenor of MPEG, Dr. Chiariglione gives the history of the Moving Picture Experts Group and explains the characteristics of the MPEG open-source software process.

For centuries my ancestors, who lived in the lower parts of the Alps near the city of Turin, had applied a simple idea: it was more comfortable for everybody if the paths criss-crossing the mountains were cobblestoned instead of left in the state in which the steps of millions of walkers had created them. It is not known whether that work was undertaken by the free decision of those mountain dwellers or by the local communal authority that imposed corvées on them during winter when work in the fields was minimal. After all, farmers are not known to be inclined to share anything with anybody and those were years in which despotism, enlightened or otherwise, ruled.

A few years ago computer people discovered that it was in (nearly) everybody's interest if the virtual equivalent of mountain paths—the raw CPU—could be “cobblestoned” with an operating system that was the result of a collective effort that could be used by all.

Traditionally, computer people have worked with data that was already represented, or could be easily converted to, a form that lent itself to processing by automatic computers. Other types of data, those that reach human ears and eyes, have a very different nature: they are intrinsically analogue. To add difficulty they are also “broadband”, a sliding definition that depends on the state of technology.

Processing and communication of audio and video data has been around for a long time but invariably as ad-hoc solutions. As part of the movement instigated by the Moving Picture Experts Group or MPEG-1, audio and video have been reduced to a form that allows the necessary process to be achieved by integrated circuits. The amount of bits have been reduced to such a level that transmission is possible over today's communication channels.

In parallel to the development of the MPEG-12, MPEG-23 and MPEG-44 standards, MPEG has developed reference software using a process similar to that of open-source software (OSS), even though the details may be frowned upon by the purists of the OSS community. It must be recognized, however, that this process had to be adapted to the rules governing the International Organization for Standardization (ISO), a traditional standards-setting organization under which MPEG operates.

The purpose of this article is to recall how digitization of audio and video was started explain the motivations that led to the establishment of the Moving Picture Experts Group summarize the elements of MPEG standards being used today and explain the characteristics of the MPEG open-source software process and the work currently under way.

The Digitization of Audio and Video

It took about 400 years after the invention of movable type, the first example of a technology for large-scale use of information processing not requiring direct human intervention, to see the invention of a technology of a similar impact. Starting in the 1830s, a long string of audio-visual information processing and communication technologies has been made available to humankind: photography, telegraphy, facsimile, telephony, phonography, cinematography, radio, television and magnetic recording.

One drawback of these technologies is that each of them has, in general, little to share with the others. Every time one of these types of information is processed, a special device has to be used. How different from the computer world where processing all kinds of information is made using the same basic technology!

The theoretical groundwork to achieve the goal of unifying all types of audio-visual information started some 15 years before the first electronic computer was built. It was discovered that a band-limited signal (of bandwidth B) could be sampled with a frequency of 2B and reconstructed without error. The second step of the groundwork was achieved some 20 years later with the definition of bounds to quantization errors depending on the number of bits used and the signal statistics.

Even though Bell Laboratories, where the theoretical groundwork had been done, made the first step of converting this groundwork into something practical with the invention of the transistor, there was still a long way to go for practical applications. Even a “narrowband” signal like speech that occupies the

0.3-3.4KHz band on the telephone wire, if sampled at 8KHz with 8 bits per sample, produced the staggering (for that time) value of 64Kbps.

After 15 years of experiments, bits were ready to play a role in speech communication. In the 1960s the then CCITT (now ITU-T) adopted a recommendation for the digital representation of speech. (This actually defined two such representations, called µ-law and A-law.) Both had a sampling frequency of 8 KHz, but the quantization law was 7 bits per sample for µ-law and 8 bits per sample for A-law, both nonlinear to take into account the logarithmic nature of human ear perception. One should not, however, attach too much meaning to this digitization of speech. The scope of application was the trunk network where multiplexing of telephone channels was more conveniently done in digital than in analogue. Nothing changed for the end users.

More interesting was Group 3 facsimile (Gr. 3 fax). An A4 page scanned by the 1728-sensors CCD of Gr. 3 fax in fine resolution mode (same resolution horizontally and vertically) holds about 4 Mbps. With the “high speed” modems of that time (9.6 Kbps) it would have taken about 20 minutes to transmit a page, but a simple compression scheme (sending “run lengths” encoded with variable-length code words instead of all blacks and whites and some bidimensional extensions) brought down transmission time to two minutes six seconds.

Digitized speech was an effective transmission method for the trunk network, but the local access remained hopelessly analogue. The advent of ISDN in the 1980s prompted the development of standards for speech compression with the bandwidth of 7 KHz, sampled at 16 KHz with a higher number of bits per sample (e.g. 14) than µ-law and A-law. Compression was needed because this kind of speech would generate in excess of 200 Kbps. Reduction to 64 Kbps and below (compression ratio of about four) was possible preserving high speech quality. This device used DSPs (Digital Signal Processing) but never gave rise to a mass market. Video presented a bigger challenge if one considers that its bandwidth is three orders of magnitude more than that of speech and involves more than one signal. Digital television is obtained by sampling the video luminance Y at 13.5 MHz and the two chrominance differences R-Y and B-Y at 6.75 MHz with 8 bits/sample. The total bitrate of 216 Mbps could be reduced to about 166Mbps by removing the nonvisual samples. Such high bitrates were unsuitable for any practical transmission medium and were used only for the digital tape (so-called D1) and transmission in the studio.

The first attempt to apply bitrate reduction to reduce this high bitrate to 1.5-2 Mbps to fit the American and European speech multiplexers of 24 and 32 digital speech channels, respectively, was (and still largely is) considered too challenging. Therefore, the input bitrate was first reduced by 2:1, subsampling the video signal in the horizontal and vertical (actually temporal, as the video signal is interlaced) directions and by further subsampling the chroma differences. Then, two simple techniques called DPCM and conditional replenishment were used. A second generation of codecs, using more sophisticated algorithms (DCT [Discreet Cosine Transform] and motion compensation), provided acceptable quality at 384Kbps and, by further 2:1 subsampling the video signal in the horizontal and vertical directions at 64/128Kbps, the bitrate of ISDN.

Going back to audio, in the early 1980s Philips and Sony developed the Compact Disc, a read-only digital storage device that employed laser technologies (a comparable system was developed at about the same time by RCA, but was short-lived). This was designed with stereo music in mind: two audio channels sampled at 44.1KHz with 16 bits per sample for a total bitrate of 1.41Mbps.

Lastly, in the US (through the Advanced Television initiative) and in Europe (through the development of an industrial company) steps were taken toward the development of a market for digital high-definition television.

The First MPEG Standards

My work experience has been in a telecommunications research establishment. The telecommunication industry used to be characterized by considerable innovation in the network infrastructure where investments were not spared and by reluctance to invest in terminal equipment. This was in part because terminals were alien to its culture (even though the more enlightened individuals were aware that unless there were new digital terminals there would not be much need for network innovation), and in part because the terminal was technically and legally outside of its competence. The attitude was “Let the manufacturing industry do the job of developing terminals.” Unfortunately, the telecommunications manufacturing industry, accustomed to being pampered and running the risk of fewer orders from the telcos based in solid CCITT standards, had no desire to make investments in something based on the whim of end users they did not understand. The consumer electronics industry, which knew end users better and was accustomed to make business decisions based on their judgment of the validity of the products, still considered telecommunications terminals out of its interest. This explains why, at the end of the 1980s, there was virtually no end-user equipment based on compression technologies, with the exception of facsimile. To make cheap and small terminals one would have needed ASICs (Applications Speciftc Integrated Curcuits) capable of performing the sophisticated signal processing functions needed by compression algorithms.

I saw the attempts being made by both Philips and RCA in those years to store digital video on CDs for interactive applications (called CD-i and DVI, respectively) as an opportunity to ride on a mass market of video compression chips that could be used for video co-communication devices. What was required was the replacement of the laborious and unpredictable “survival-of-the-fittest” market approach of the consumer electronics world with a regular standardization process.

MPEG-1

So started MPEG in January 1988 with the addition of the mandate a few months later for audio compression and the function needed to multiplex and synchronize the two streams (called “systems”). In four years the first standard MPEG-1 was developed. Interestingly, none of the two original target applications—interactive CD and digital audio broadcasting—are currently large users of the standard (video communication has not become too popular either). On the other hand, MPEG-1 is used by tens of millions of video CDs and MP3 players. One feature of MPEG-1 that is remarkable: MPEG-1 was the first audio-visual standard that made full use of simulation for its development. The laboratory at which I worked took part in the development of the 1.5-2Mbps video conference codec using three 12U racks and minimal support from computer simulation. Even more significant for future implications was the fact that MPEG-1—a standard in five parts—has a software implementation that appears as “part 5” of the standard (ISO/IEC 11172-5).

MPEG-2

In July 1990, MPEG started its second project, MPEG-2. While MPEG-1 was a very focused standard for well-identified products, MPEG-2 addressed a problem everybody had an interest in: how to convert the 50-year-old analogue television system to a digital compressed form in such a way that the needs of all possible application domains were supported. This was achieved by developing two system layers. One, called the MPEG-2 Transport Stream (TS), was designed for error-prone environment targets (such as cable, satellite and terrestrial) of the transmission application domains. The other, called MPEG-2 Program Streams (PS), was designed to be software-friendly and was used for DVD. The idea was that MPEG-2 would become the common infrastructure for digital television; indeed, something that has been successfully achieved if one thinks that at any given moment there are more bits carried by MPEG-2 TS than by IP. The title of the standard “Generic Coding of Moving Pictures and Associated Audio” formally conveyed this intention. By the time MPEG-2 was approved (November 1994), the first examples of real-time MPEG-1 decoding on popular programmable machines had been demonstrated. This was, if there had been a need for it, an incentive to continue the practice of providing reference software for the new standard (ISO/IEC 13818-5).

MPEG-4

In July 1993, MPEG started its third project, MPEG-4. The first goal is reflected in the original title of the project, “very low bitrate audio-visual coding”. Even though no specific mass-market applications were in sight, many sensed that the digitization of narrowband analogue channels, such as the telephone access network (Internet was not yet a mass phenomenon), would provide interesting opportunities to carry video and audio at a bitrate definitely lower than 1Mbps, roughly the lowest bitrate value supported by MPEG-1 and MPEG-2. For that bitrate range it was clear that a decoder could very well be implemented on a programmable device, unlike other MPEG standards. It was possible that there would eventually be more software-based than hardware-based implementations of the standard. This was the reason the reference software, part 5 of MPEG-4 (ISO/IEC 14496-5) has the same normative status as the traditional text-based descriptions of the other parts of MPEG-4.

MPEG-4 became a comprehensive standard as signaled by its current title, “coding of audio-visual objects”. The standard supports the coded representation of individual audio-visual objects whose composition in space and time is signaled to the receiver. The different objects making up a scene can even be of different origins: natural and synthetic.

This does not mean, however, that a particular implementation of the standard is necessarily “complex”. An application developer may choose among the many profiles—dedicated subsets of the full MPEG-4 tools—to select the one used to develop his application. For all these reasons, it is expected that MPEG-4 will become the infrastructure on top of which the currently disjointed world of multimedia will flourish.

Why a Standard for MPEG-4?

Readers may wonder why, if the coding algorithm is implemented in software, there was a need to develop a standard. Shouldn't it suffice to download the code that allows for the decoding of the particular algorithm used to produce the bitstream of your interest?

In the early days of MPEG-4's development this question used to be asked very often, but today, with the ever-expanding use of MP3, it is easier to understand the benefits of having a standard: a playback device is not necessarily connected to the network. Instead, it may be on a broadcast channel, a stand-alone or portable device; the devices can use many different CPUs for which it could be too costly to develop playback codes; the hardware may use an ASIC for the audio-visual decoding that is not upgradeable; or it may have been designed to run with just the amount of RAM that the standard algorithm requires. In other words, it is simpler to have a common standard on which business opportunities can multiply, instead of having to struggle with incompatibilities all over the place.

Lastly, it should be kept in mind that compression coding is not a transparent operation. In general, the lower the bitrate used, the more the quality is affected negatively. Transcoding from one algorithm to another may simply produce garbage. Also, the idea that compression technology keeps improving is a myth. Only now, after many years, is MPEG re-issuing a call for proposals for video compression technologies because of the feeling that there may be something worth considering. For audio compression MPEG is still at the level of issuing a call for evidence because the group is not convinced this is an area currently worth pursuing.

The MPEG-4 “Open Source”

The very size of the standard has transformed the development of the reference software into a huge undertaking. It is therefore interesting to see how such a project was managed. These are the most important features:

- The condition was set that any component of the standard, both normative (decoder) and informative (encoder), had to be implemented in software. For any proposal to be accepted and adopted, it was a condition that source code be made available and the copyright released to ISO.
- For each portion of the standard, a manager of the code was appointed: a representative of Microsoft and MoMuSys for video in C++ and C respectively, Fraunhofer for natural audio, MIT for Structured Audio, ETRI for Text-to-Speech interface, Optibase for the so-called “Core” (the code portion on which all media decoders and other components plug in), Apple for the so-called MPEG-4 File Format, etc.
- Each portion of the standard had a manager of experiments appointed. This manager integrated the code of the accepted tools in the existing code base.
- Unlike traditional open-source software projects, only MPEG members could participate in the project. Discussions were usually held (and the practice still continues) on e-mail reflectors that are open to non-MPEG members.

MPEG is a place where new ideas are continuously forged. One idea was generated by the fact that while the reference code is intended to be “reference” (normative or informative as the case may be), it is not intended to be efficient. Therefore, since December 1999, MPEG has been working on a new part of MPEG-4 that will contain optimized code (e.g., optimized ways to search for motion vectors, a computationally expensive part of the standard). Any implementer can take this code and use it free of copyright. The condition has been set, however, that such optimized code should not require patents. A second idea, launched in October 2000, led to the decision to develop an MPEG-4 “reference hardware description”. It is expected that this will further promote the use of MPEG-4 as the basic multimedia infrastructure in both software and hardware.

The text of the so-called “copyright disclaimer” that is found on all MPEG-4 software modules is given below.

> This software module was originally developed by <First Name 1> <Last Name 1> (<Company Name 1>) and edited by <First Name 2> <Last Name 2> (<Company Name 2>), <First Name 3> <Last Name 3> (<Company Name 3>), in the course of development of the <MPEG standard>. This software module is an implementation of a part of one or more *<MPEG standard>* tools as specified by the <MPEG standard>. ISO/IEC gives users of the <MPEG standard> free license to this software module or modifications thereof for use in hardware or software products claiming conformance to the <MPEG standard>. Those intending to use this software module in hardware or software products are advised that its use may infringe existing patents. The original developer of this software module and his/her company, the subsequent editors and their companies, and ISO/IEC have no liability for use of this software module or modifications thereof. Copyright is not released for non-<MPEG standard>-conforming products. <Company Name 1> retains full right to use the code for its own purpose, assign or donate the code to a third party and to inhibit third parties from using the code for non-<MPEG standard>-conforming products. This copyright notice must be included in all copies or derivative works. Copyright (199_).

Today and Tomorrow

Currently, MPEG is engaged in the final stages of development of MPEG-7, “Multimedia Content Description Interface”, a standard to describe audio and video information, be it at the level of a complete movie or as a single object in a picture. The standard will be approved in July 2001. Also, for this standard there is a huge body of reference code that has been developed according to rules similar to those of MPEG-4.

In June 2000, MPEG started a new project called MPEG-21, “Multimedia Framework”. In this context, MPEG will develop and integrate, in collaboration with other bodies, all the technologies that are needed for electronic commerce of digital content on the network.

The key technologies that are needed by this project are:

1. Digital Item Declaration: a uniform and flexible abstraction and interoperable schema for declaring Digital Items.
2. Content Representation: how the data is represented as different media.
3. Digital Item Identification and Description: a framework for identification and description of any entity regardless of its nature, type or granularity.
4. Content Management and Usage: the provision of interfaces and protocols that enable creation, manipulation, search, access, storage, delivery and (re)use of content across the content distribution and consumption value chain.
5. Intellectual Property Management and Protection: the means to enable content to be persistently and reliably managed and protected across a wide range of networks and devices.
6. Terminals and Networks: the ability to provide interoperable and transparent access to content across networks and terminal installations.
7. Event Reporting: the metrics and interfaces that enable users to understand precisely the performance of all reportable events within the framework.

Of particular interest for this article is item five, Intellectual Property Management and Protection. Since MPEG-2 times, MPEG has been mindful of the need to provide solutions for those content and service providers who attach monetary value to content. So far, the solutions provided by MPEG have been at the level of enabling the use of proprietary protection technologies. However, these have the disadvantage that consumption of protected content is no longer transparent to the user, even in the case where users are willing to adhere to the conditions set by the rights holder. This is the reason MPEG is now developing a solution to provide “interoperability at the level of protected content”.

Patents in MPEG Standards

In the 15th century, “Letter patents” were already in use in Venice and Florence but unknown in Mainz. Therefore, the only way for Johannes Gutenberg to protect his invention was by hiding the secrets from everybody, including his financial backers, and this eventually led him to ruin. In the 19th century, all audio-and video-related inventions were protected by patents. This continued in the 20th century although the centre of gravity progressively shifted from individuals to the companies hiring them. When the prospects of using digital technologies became clear, all companies and organizations started making or funding research in audio and video coding. Today, the number of patents is counted by the thousands.

When MPEG started its work in audio-visual coding, it was immediately evident that either MPEG play by the existing rules in the audio-visual world—that standards usually require patents for their implementations—or it would have been impossible to produce any standard of practical value. This is in addition to the difficulty MPEG, with no funds of its own, faced becoming aware of patents required to implement its standards.

The problem of patents in standards is, of course, well known to the three main international standards organizations: IEC, ISO and ITU. They have developed the following general policy:

1. No patent should be required to implement a standard or;
2. The rights holder should release the rights or; and
3. The rights holder should make a statement where he or she engages to give license to his patent "on fair and reasonable terms and nondiscriminatory conditions".

MPEG has, therefore, developed a policy for the development of its standards that deliberately neglects consideration of patents and seeks only to achieve the optimum performance. The result has been that MPEG standards usually require a large number of patents.

As many as 100 different standards are reportedly needed to implement an MPEG-2 decoder. Because of the high interest in a "one-stop shop" for MPEG-2 patents, a private organization giving license to most MPEG-2 patents has been set up. Interestingly, the amount to be paid for the patents in an MPEG-2 decoder has remained constant, while the number of relevant patents has increased.

The same is happening with MPEG-4. The MPEG-4 Industry Forum (http://www.m4if.org/) has been established with the goal of kicking off patents pools for MPEG-4 profiles. Of course, the MPEG-4 case is much more complex since many business models require decoder download. A similar organization for MPEG-7 is likely to be set up soon.

Conclusions

Through a completely different process, MPEG—as representative of the world of audio and video—has come to a conclusion similar to the world of data-processing regarding the need to provide open solutions expressed in software (or, as the case may be, hardware) to technologies that are considered part of the "infrastructure". The outstanding difference is that while the data processing world likes to define fully open technologies, MPEG bows to the reality of the world of digital audio and video where patents are found all over the place. Therefore, reference software (and reference hardware description) is copyright-free but, in general, not patent-free.

MPEG-21, a project to define an ecosystem of content on the network, places standardization of the infrastructure one level higher compared to what has been done so far. As the provision of reference software, be it normative or informative, is now an integral part of MPEG standards, it can be expected that considerable challenges lie ahead when MPEG will need to accommodate libertarian spirits with other, more mundane considerations. But, I believe it is better to deal with this problem in a group of technical experts than in a court of law or in a parliament.

The cooperation of all parties is sought.

Resources

Leonardo Chiariglione was born in Almese (Italy). In 1971, he joined CSELT, the corporate research centre of the Telecom Italia group, where he is head of the Television Technologies Research Division. He originated the ISO MPEG (Moving Pictures Experts Group) standards group, of which he is the Convenor, in 1988. The next year he began Image Communications, a EURASIP journal for the development of the theory and practice of image communication, of which he is the editor-in-chief. In 1994, he originated the Digital Audio-Visual Council (DAVIC) where he served as president and chairman of the board until 1995. In 1999, he was appointed Executive Director of the Secure Digital Music Initiative (SDMI) to develop specifications enabling multiple business models of electronic commerce of secure digital music.

Archive Index Issue Table of Contents

Advanced search

Running Linux with Broken Memory

Rick van Rein

Issue #83, March 2001

Being able to run Linux flawlessly on a machine with faulty memory that would otherwise be discarded makes a lot of sense—the BadRAM patch makes it happen.

There are several common causes for failure of memory modules. Before solving the problems in software, let's turn to these causes. To understand them, please take a look at Figure 1, which shows schematically what the structure of a (faultless) memory is. Note how the memory is laid out in a (roughly equal) number of rows and columns. Each crossing marks the location of a memory cell, usually storing a single bit. As a result, a memory cell's address is the combination of its row address and column address. Memory modules are fed with these two halves of the address separately (sequenced), so that their address bus is only half the size you would expect. This halving of addresses is handled a by the hardware on your motherboard.

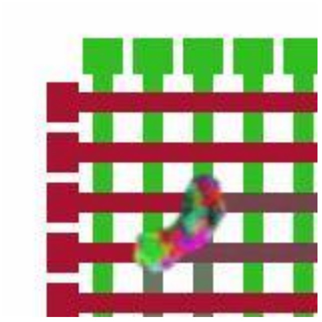


Figure 1. Memory Structure without Problems

The first place where things can go wrong is during the manufacture of these memory modules. This is a very sensitive process, mainly because the dimensions of the chip layouts are getting smaller as the world grows hungry for more memory. Chip manufacture is also quite wasteful of environmental resources because a lot of highly purified chemicals are needed to construct the tiny, sand-derivative product that you eventually buy. Moreover, the sensitivity of the chip manufacturing process makes many chips fall out due to error.

One method of partially solving this problem is to build redundancy into the chips, say including 33 rows of memory cells for every 32 rows; when one row fails, route the signals around it to the extra row. Indeed, this technique is applied by some manufacturers. Nevertheless, unsustained rumors claim there still is a 60% drop-out rate. Needless to say, this factor makes memory expensive.

Errors caused during production can take the shape of a speck of dust that sat on the chip while enlightening or developing one of its layers, as in Figure 2. The grayed-out area is the part that may malfunction as a result of that speck of dust.

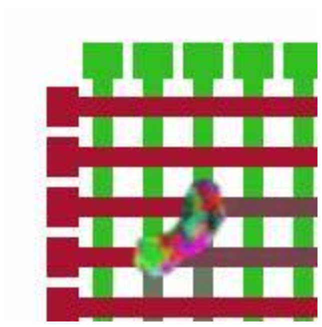


Figure 2. Memory Etching Disturbed

Another source of error is static discharge. This is the same effect that makes a woolen or nylon sweater spark a shock when taken off. The reason for the spark is a high voltage between two nearby surfaces. When they come close enough, the spark suddenly jumps over, ionizing the air it passes and, thus, suddenly making it conduct quite well, resulting in a high current for a very short period. Actually, this is the same effect as lightning, only not as powerful, of course. The small structures in a chip are quite sensitive to these aggressive discharges. Static discharge usually damages the “buffers”, the connections between rows/columns and the address selection logic, as shown in Figure 3. This effectively means that a whole row, or a whole column, or even a few of them, become unusable, shown again by the grayed-out area. There are the memory fault patterns that I have seen the most.

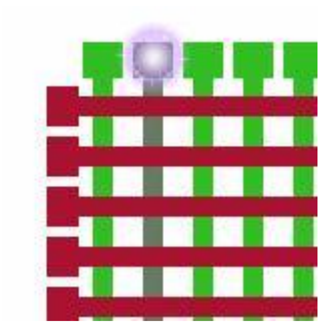


Figure 3. Memory Buffers Damaged by Static Discharge

A last source of error, and up to now I have no certain proof of ever having encountered it, is the gradual decay that all chips are subjected to. This is the process by which the sharply distinct regions of different materials on the chip blur and mix together. This is a natural effect, and if a chip is kept sufficiently cool, it usually takes decades to happen. In the case of memory, no problem, it usually gets outdated for its speed and size long before this point.

The important point of all these errors is that they stick—I am not aware of any technical cause that could lead to random errors. Mind you, it may happen that errors only occur in particular patterns of the surrounding bits, but even then, the error once settled, always occurs in that situation.

Furthermore, errors don't happen for no reason. In all of the above situations, a sudden burst of energy causes the errors, or they are caused by a very slow process. So, we need not expect errors to jump up all of the time, not even on a defective memory module.

Because errors in memories do not evolve, it is usually possible to circumvent the errors by making smart use of dynamic memory allocation techniques.

Detecting Memory Errors

There are basically two ways of detecting errors in memories. One approach is the use of a special program that scans all of the memory in the machine and reports the failing addresses. One such program that excels in finding errors is **memtest86**, targeted at i386-based machines. Naturally, trusting the results from such a memory checker relies on the stability of memory errors, which usually turns out to be an okay assumption; I've been running my machine for months without a change in the detected error patterns.

The alternative is based on hardware detection. In the 30-pin SIMM days, it was quite common to include a ninth bit with “parity” information on the memory module. The idea is that the ninth bit, which is actually stored separately on the memory module, contains the bit to make the parity of the whole chip even (or odd, whatever the design of the motherboard wants). When writing, this bit is generated, and when reading back, it is checked. If this read-time check fails, a parity error is thrown.

A modern alternative to parity bits is ECC, short for Error Correction (and detection) Code. These are usually based on some CRC hash from the bits stored, and they can be used to detect up to three faulty bits out of 32, as well as to correct up to two faulty bits. (I am actually uncertain about these precise values, but it's the principle that counts.) So, with ECC RAM, it is possible to detect errors, as well as correct them if they are not too savage.

Modern memory modules as we use them in PC workstations usually come with neither parity nor ECC. On the other hand, it is standard to find ECC in high-end (server) systems as sold by VA Linux and Sun. These usually work with ECC memory, or at least with parity memory. I believe that most common PCs are capable of working with ECC memory, they just don't get that memory plugged in because ECC RAM is more expensive. Yes, this is the low-quality world of PCs blinking an eye at you.

With parity RAM or ECC RAM it would be possible to detect errors on the fly, without the need for a special program like memtest86. Currently however, the BadRAM patch we present here is kept as simple as possible (to increase chances of acceptance in mainstream kernels) and, therefore, does not deal with the added possibilities of ECC RAM.

Describing Errors as Memory Patterns

We assume a memory checker like memtest86 finds errors in memory modules. As explained, this checker lists all erroneous addresses it finds. And, because a whole row or column may fail after a static discharge, this could lead to extremely long lists of errors, hundreds of errors not being an exception. Naturally, such a list would be tedious and error prone to enter into the kernel, and it may also give rise to problems caused by strictly limited resources at boot time. It would be ideal to have a very small representation of all the errors found.

Luckily, memory errors are usually laid out in regular patterns, such as starting at address **0<\#215>1234**, every **0<\#215>0040** bytes, over 16 occurrences—or a bit more variation, of course.

The regularity is often easiest to see when doing binary. That is because the rows and columns are addressed by spreading address bits over these lines, and by interpreting part of the address to decide whether the used address lines fall into the correct region.

A single error can be described as an address, of which all bits are valuable information. We write this, for example, as **0<\#215>1234,0<\#215>ffff** where the first number is the *base address* and the second number is a *mask*. The “1” bits in the mask indicate which of the corresponding base address bits are valuable. If, aside from this address, also the address plus **0<\#215>0040** is wrong, then we can simply alter the mask to represent this. The resulting address/mask pair now becomes **0<\#215>1234,0<\#215>ffbf**. We can see that this is correct because the addresses covered are all addresses A for which (A &

0<\#215>ffbf)=**0<\#215>1234**, or concretely, **0<\#215>1234** and **0<\#215>1274**. If there are 16 faulty addresses with this intermittent offset, then the whole thing becomes **0<\#215>1234,0<\#215>fc3f** to capture faulty addresses: **0<\#215>1234**, **0<\#215>1274**, …, **0<\#215>15f4**.

This approach works well to capture faults over a row or column, as long as these cover a power of two for bits, which is normal. It also works to capture the specks of dust that disable a few neighboring bits on the chip. But what if a memory module contains more errors? Or, similarly, if multiple memory modules each have errors? To cover these cases, we usually assume a list of the aforementioned address/mask pairs. In practice, it turns out that five of these pairs suffice for most practical situations. Be aware that any set of errors can always be compacted in as little as one such pair, albeit with loss of good addresses, **0<\#215>0000,0<\#215>0000** being the ultimate example that captures all errors, but unfortunately, no good addresses. With five pairs, we usually have no problems that extreme.

memtest86, starting with version 2.3, is able to generate BadRAM patterns that can directly be entered on the command line of a BadRAM-patched Linux kernel. In the above example, memtest86 would report a sequence of evolving patterns, eventually leading to:

```
badram=0<\#215>1234,0<\#215>fb3f
```

Having written this down, you may now reboot the system and enter this on the command line:

```
LILO: linux badram=0<\#215>1234,0<\#215>fb3f
```

and your system should boot fine, ignoring the broken memory parts. Now, go ahead and add this line to your /etc/lilo.conf:

```
append="badram=0<\#215>1234,0<\#215>fb3f"
```

and run LILO. You will not have to enter the address/mask pairs anymore on consecutive boots.

Patching the Kernel

My old ZX Spectrum had a nifty way to expand memory with an additional 32K. The Sinclair memory expansion kit was comprised of 64K chips, of which either the high or the low half was known to work while the other half was faulty, which made the price lower than the “proper” expansion method with 32k chips. By selecting some toggle on the motherboard, the computer was instructed which half of the expansion memory chips it should use. Basically, the BadRAM patch does the same thing, with a bit more refinement.

Memory management units, which are a necessity for all Linux distributions to work well, redirect a page as accessed by a “user space” program to any physical page. What appears like a long stretch of memory allocated just for your user process is, in reality, scattered over the memory modules (and may even be swapped out). When a user-level program allocates memory, the kernel gives it out by allocating single pages of physical memory.

The only part of Linux that works in terms of physical memory addresses is the kernel. It is loaded in the beginning of the memory (obviously, you must not have errors in that region) and may allocate blocks of memory from the same pile of free memory that is also used to serve the user process.

This pile of memory is filled with the physical pages at boot time. Actually all that the BadRAM patch does is leave out those pages that fall in one of the address/mask pairs entered on the command line.

This means that all the work involved in BadRAM is done at boot time. Afterward, the only effect is that the pile of free memory is a tad smaller. Extensive benchmarks have shown that this has no measurable influence on runtime performance.

Alternative Applications

When you make a patch like this, and certainly when you're “SlashDotted” over it, you receive a lot of interesting e-mail, some with ideas even weirder than those I thought of.

One proposal I've often heard is to perform a memory test at boot time. Although this may seem interesting, it is not practical. memtest86 does an excellent job, but it requires several hours. You don't want this at boot time, and you also don't want to have a bad memory test (we already have that in most PC BIOSes, anyway, and BadRAM modules usually pass that test). The option of making a LILO boot alternative for memtest86, or of having a memtest86 boot floppy, has always been my preference.

I have had some reports of errors in motherboards which corrupted a particular address of physical memory, perhaps due to a short circuit with an on-board peripheral. One such reporter informed me that he had put four modules of 512M in his machine to limit the chances of hitting the erroneous address, but his problems were entirely resolved by throwing out a single memory page with the BadRAM patch. Discarding 4K out of 2G made his machine work flawlessly.

I have also talked with someone who owns a PC from a large PC-at-home project. The rather proprietary architecture of his Compaq Deskpro did not foresee an option to switch off the 15M-16M memory hole needed for some ISA cards. So, expanding memory to 24M did not work on Linux 2.2 because setting mem=24M means that the region 15M-16M works as, you guessed it, a memory hole. But, after adding **badram=0<\#215>00f00000,0<\#215>fff00000** to inform the kernel that the hole should be treated as BadRAM, he got his additional memory going and was ready to add even more.

Finally, I received romantic responses from people that had worked with older systems that detected memory faults (using either parity schemes or ECC schemes) and assigned a lower level of trust to such faulty pages. When a page was “under evaluation”, it would, at most, be used to load program code, the idea being that program code is a verbatim copy off a hard disk and can be restored if the error pertains. If it worked well for a while, the error apparently had been something spurious, and the page would be upgraded to “trustworthy” again. However, if program storage also didn't work, the page would be put out of use. This scheme would be very interesting to support, but it would take up CPU cycles at runtime and, therefore, should be seen as a very fancy feature.

Possible Future Extensions

There are a few ways to extend upon the BadRAM efficiency. It is certainly possible to exploit ECC modules, both the ones that have recoverable and unrecoverable errors. Since I lack both types of modules, unfortunately, this is beyond my current ability. Also, it burns CPU cycles and falls into the category of fancy features.

Another option would be to exploit the slab allocator in the kernel. Slabs are small, uniform and reusable memory blocks that the kernel allocates in arrays that span, as closely as possible, an integer number of memory pages. It would be possible to exploit the error address information in more detail by using BadRAM pages for slabs, thereby avoiding the allocation of slabs that overlap error addresses. Ideally this could reduce the memory loss to absolutely zero. In practice, however, it will not be far off the standard BadRAM performance because an average system does not use many slab pages at all. For this reason, I doubt if the additional CPU overhead and coding effort would be worthwhile.

It is my sincere hope that memory marketing companies will pick up this idea and start to publish (cheap) memory modules based on broken memory. I propose a schema to classify such modules with a logarithmical degree of “badness” as part of the in-kernel documentation on the BadRAM patch.

All that remains now is to wish you good luck with your chase for broken memory. Perhaps some befriended user of a less mature operating system could spare it.

Resources

Rick van Rein is a PhD student in computer science at the University of Twente. He closely follows developments in the GNU community, which usually inspires him in his daily work. The BadRAM patch is his way to donate to the GNU community.

Archive Index Issue Table of Contents

Advanced search

Introducing SOAP

Reuven M. Lerner

Issue #83, March 2001

SOAP is something you may find a use for, even if you're not intersted in three-tier web applications.

In the January and February installments of “At the Forge”, I demonstrated a simple three-tier web application using a database, web server and the Mason templating system for mod_perl. We were able to see some of the advantages and disadvantages of a three-tier web application, particularly when compared with its two-tier counterpart.

But as I pointed out last month, our three-tier architecture was incomplete and wasn't necessarily a fair demonstration. That's because our Perl middleware object layer had to reside on the same computer as the components we wrote for HTML::Mason, a templating system built on mod_perl. Depending on how you count things, this might be considered a two-tier application, albeit one with an object-oriented abstraction layer between the tiers.

In order to put the Mason components and Perl objects on separate computers, we somehow need the ability to call an object method across a network. That is, the following line of Perl would work, regardless of whether $object resides on the same computer as our Apache server or somewhere else on the Internet:

```
$object->method($arg1, $arg2);
```

Distributed-object technology and remote-procedure calls have existed for many years on a variety of platforms. In almost every case, this technology was restricted to a particular language or platform. DCOM (Distributed Component Object Model) allows objects of any language to communicate but only under Windows. Java's RMI (Remote Method Invocation) can only communicate with other Java objects. CORBA is an exception to this, allowing objects to communicate across platforms and languages, but CORBA is complex, has taken awhile to get off the ground and isn't yet a part of most programmers' knowledge base.

In response to these proprietary and complex protocols, a number of people in the Internet community have created SOAP, the Simple Object Access Protocol, that makes it extremely easy to create distributed applications. Two of the biggest proponents of SOAP have been Dave Winer (famous for his Scripting News “weblog”) and Microsoft, which is not usually associated with open standards and cross-platform protocols. Regardless of what we in the Linux community might think, Microsoft has publicly embraced SOAP, making it a cornerstone of its .NET effort.

SOAP History and Concepts

SOAP depends on the idea that any two computers on the Internet can communicate using HTTP, the protocol that powers the Web. (Actually, SOAP can be transmitted over nearly any high-level protocol, including SMTP and POP3, but HTTP is by far the most common.) It then transmits information using XML, the markup language that allows us to create tags and document standards. The server turns the incoming XML into an object method call, and then turns the object's response into an XML document that is returned as the HTTP response. Since both HTTP and XML are open standards, published by the World Wide Web Consortium, they can be (and are) implemented on a variety of platforms and, thus, interact without any trouble.

The predecessor to SOAP, known simply as XML-RPC, provided a simple mechanism for remote procedure calls (RPC) using data formatted in XML and transmitted over HTTP. For a variety of reasons, including the fact that XML-RPC could not handle advanced data structures, the W3C adopted SOAP.

A number of languages and platforms continue to support XML-RPC, and it's possible that some situations might call for its use because it has a smaller overhead. Practically speaking, however, the fact that SOAP has gotten so much attention has led to the development, use and debugging of its libraries to a much greater extent than those for XML-RPC. As of this writing, however, there are more implementations of SOAP than XML-RPC, meaning that your choice of platform or language might force your hand toward one protocol or the other.

SOAP, as its name implies, expects to work with objects rather than simple procedure calls. Thus, SOAP client invokes a method on a particular object on the server. The method is specified in the body of the XML document itself, while the object with which it is associated is named in an HTTP “SOAPAction” header. Of course, we also need to specify a computer name and port to which the SOAP request can be directed.

The server itself, including its name and the port number on which the SOAP request is transmitted, is known as the SOAP proxy. This makes sense when you consider that the HTTP server is simply relaying an object method invocation and isn't doing any of this work by itself. Do not confuse the SOAP proxy with an HTTP proxy. An HTTP proxy relays requests from an HTTP client to an HTTP server and often performs security checks and caching. A SOAP proxy, by contrast, relays messages between a SOAP client and an object on the proxy's computer.

The object for which the SOAP server acts as a proxy is sometimes known as the endpoint and is specified in a “SOAPAction” HTTP header. The name of the endpoint can be virtually any text string, including hierarchy separators such as :: and /. In practice, the endpoint has a direct connection to the object hierarchy associated with the language in which the SOAP proxy is written. In Perl, the endpoint might be something like “Foo/Bar”, which refers to the Foo::Bar object located in the file Foo/Bar.pm.

SOAP Anatomy

Let's look at a simple SOAP conversation. Our examples will demonstrate SOAP on top of HTTP, which is the most common configuration. There may be slight differences when working with other protocols.

HTTP is stateless, meaning that every connection between two computers consists of one request (from the client to the server) and one response (from the server back to the client). The request and response are each divided into two parts, known as the headers and the body. Of course, the client and server can add any other headers they want, opening the door to all sorts of specialized communications protocols.

The body of a SOAP request or response will be in XML. (If you have never worked with XML before, don't worry; while it can be a deep and intriguing topic, you don't need to know much XML to work with SOAP.) Each SOAP message—a request or response—consists of an optional SOAP header and a mandatory SOAP body wrapped inside of a SOAP envelope. The envelope identifies the contents as belonging to SOAP and sets out the namespaces that will be used for the rest of the message. The headers describe the data in the body, and the body contains the method call or its results.

In order to invoke an object on a remote server via SOAP, we will have to open an HTTP connection to the appropriate URL, identifying the object via the SOAPAction header. We send an XML document containing a SOAP envelope, inside of which our SOAP headers and body identify the method to be invoked on this object, as well as any parameters that the method might require. The client must additionally be prepared to parse the response returned by the

SOAP server, extracting data structures contained in that response and using them as necessary.

A SOAP server performs complementary actions, receiving the SOAP request, parsing its contents and invoking the appropriate method on the local computer with the passed parameters. The server also returns the SOAP response to the client, containing one or more values as necessary.

Now that you understand the terminology associated with SOAP, you can forget nearly all of it. SOAP implementations provide us with an abstraction layer that allows us to ignore the fact that it communicates via HTTP and that the request and response use XML. When your program invokes a remote object method using SOAP, it cares about receiving a response; the way in which the request and response are packaged is of little concern.

Our Back-End Object

I'm going to write some demonstration programs in Perl using the excellent SOAP::Lite module written by Paul Kulchenko. This should give you some idea regarding how to write SOAP clients and servers, as well as how to integrate them into your web applications. Despite its name, SOAP::Lite offers a rich array of functionality and can be a great way to add SOAP functionality to your Perl programs. Similar SOAP libraries and objects are available for most major programming languages, so don't think that SOAP is available only for Perl.

Because SOAP acts as a proxy for an object, we first have to create an object whose methods will be available over the network. Listing 1 contains the simple “Text::Caps” Perl module, which handles two fairly useless methods:

Listing 1. Caps.pm, the Perl Module

- **capitalize**, which takes a single string as a parameter and returns the capitalized version of that string
- **capitalize_array**, which does the same thing as capitalize, but to each element in a list of strings rather than to a single string

Notice that while SOAP describes everything in terms of objects and methods, this sample module uses standard Perl modules and subroutines rather than object-oriented syntax. So, when I mention the capitalize method for the Text::Caps object, I really mean that we'll be invoking the Text::Caps::Capitalize subroutine.

A Standalone SOAP Server

SOAP is normally carried over HTTP, which sits on top of TCP/IP. This means that we can create a simple SOAP server by taking advantage of the TCP/IP socket code that comes with Perl. However, SOAP::Lite does much of the dirty work for us; we don't have to create the socket or wait on it. Rather, we create an object of type SOAP::Transport::HTTP::Dæmon, which knows how to act as the appropriate kind of SOAP server. You can see the source code for such a simple server in Listing 2.

Listing 2. A Simple Standalone SOAP Server

The code is relatively simple yet will look odd to even the most experienced Perl programmers. That's because objects associated with SOAP::Lite usually return themselves to indicate success. This allows us to invoke more than one method in a single call. In other words, we can say

```
   $object->method1()->method2();
```

rather than the traditional

```
   $object->method1();
   $object->method2();
```

You may choose to work with SOAP::Lite using either syntax, but the first version is common in documentation.

When we invoke the “new” constructor for SOAP::Transport::HTTP::Dæmon, we pass it two arguments: the name of the computer and the port on which the server should listen for connections.

Once we have created the server object, we must tell it where objects are located. This is a security feature, albeit one that can take some time to understand. Normally, Perl looks for modules in @INC, an array of directory names. When we import a module, Perl searches sequentially through each element of @INC until it finds our module. If it fails to find our module, Perl returns an error message.

However, since SOAP exposes our modules to the entire world, we must be careful before making them available. Perhaps some of our modules return confidential data or manipulate information in a relational database. In order to ensure that only those modules we wish to expose are actually available via SOAP, SOAP::Lite completely ignores @INC when handling incoming SOAP requests. Only those modules explicitly mentioned in a call to dispatch_to(), or in a directory named in dispatch_to(), will be available via SOAP.

In a sense, dispatch_to() effectively defines the equivalent of @INC for incoming SOAP requests. If a module resides in a directory not mentioned in dispatch_to(), it will be invisible to SOAP requests. That this is not the same as modifying @INC.

Note that while I use /tmp for the examples in this article, it is a poor idea to use /tmp in this way in a real development or production system. If you want to put SOAP-related Perl modules in a separate directory from /usr/lib/perl, I strongly suggest that you keep it on the main file system, such as in /usr/lib/soaplite.

Testing Our Server

Now that our standalone SOAP server is running, we should test it to see if it works. In order to do that, we must create a SOAP request, send it to the server and then parse through the XML-encoded SOAP response that it returns. Luckily, SOAP::Lite includes such a utility, SOAPsh.pl. This small program allows us to create and send SOAP requests interactively, displaying the results. SOAPsh.pl alone justifies the download of SOAP::Lite, even if you are planning to work with another SOAP library for Perl.

If we are running our standalone SOAP server on localhost (i.e., on the same computer as we run SOAPsh.pl), and if we are running it on port 8080, we can invoke it as follows:

```
perl SOAPsh.pl http://localhost:8080/ Text/Caps
```

Notice how the first argument to SOAPsh.pl is the URL of the SOAP server, and the second argument is the object that we want to invoke. You can avoid a lot of grief by remembering that the second argument must be passed using a URL-style object hierarchy divider, namely a slash (/). Typing “Text::Caps” rather than “Text/Caps” will confuse the SOAP server and result in hard-to-debug errors.

If your invocation of SOAPsh.pl succeeds, you will see the following prompt:

```
Usage: method[(parameters)]
>
```

The “>” sign indicates that it's your turn to type and you can invoke any method for the object to which you've connected. You may now call any method that the object supports, including any parameters. So to capitalize a word, I simply type:

```
> capitalize('abc')
```

Because my SOAP client and server are both on the same computer, the response is nearly instantaneous. SOAPsh.pl prints out:

```
--- SOAP RESULT ---
$VAR1 = 'ABC';
```

Hey, that's pretty great! I just invoke an object method across the network. That wasn't so hard, was it?

SOAP would be nice if we could send simple scalars back and forth. But we can send and receive a variety of data types. For example, we can invoke capitalize_array, sending a list of arguments:

```
> capitalize_array('abc', 'def', >'GHi')
```

The return value is an array reference:

```
--- SOAP RESULT ---
$VAR1 = bless( [
                'ABC',
                'DEF',
                'GHI'
                         ], 'Array' );
```

The returned array reference looks a bit funny because it has been turned into a format that SOAP::Lite can send and retrieve. We will soon see how our programs can ignore this intermediate format, seamlessly exchanging complex data structures over the Internet.

Examining the SOAP

As you can see, it's possible to work with SOAP without understanding the underlying XML-encoded data. However, debugging SOAP problems often requires that you look at the XML as well as the HTTP headers that are sent in the request and the response.

SOAP::Lite objects support the on_debug( ) method, which takes a subroutine reference as an argument. This subroutine is invoked for each SOAP transaction, meaning that we can log information to the disk or screen. The simplest use of on_debug( ) is as follows:

```
on_debug(sub{print STDERR @_})
```

In other words, we ask SOAP::Lite to send a copy of everything to STDERR. This provides us with a marvelous opportunity to see what happens behind the scenes. After we invoke this method, SOAPsh.pl reminds us that we invoked a local method rather than a SOAP method:

```
--- METHOD RESULT ---
SOAP::Lite=HASH(0x82e1174)
```

With debugging turned on, our invocation of capitalize(abc) from before gets translated into a SOAP request (see Listing 3)

Listing 3. SOAP Request

As you can see, the request is divided into a header and a body, as with all HTTP requests. And as with a normal HTTP request, we indicate an action ("POST") along with a URL, as a Content-Length (indicating the number of bytes in the request) and the Content-Type (which is always going to be "text/xml").

Then the fun begins: the final header is SOAPAction, which names the object and method that are being invoked. The SOAPAction header is designed to allow corporate firewalls to filter out dangerous objects and methods from being invoked. Currently, however, it would seem that support for SOAPAction is relatively hard to find. Besides, information about both the object and its method are buried inside of the XML request and response themselves, making the header unnecessary for parsing purposes.

The XML itself begins with an XML declaration and then a SOAP envelope. Inside the envelope is an optional header (not shown in this particular invocation) and a mandatory body. The body names the object and method that we wish to invoke, as well as any arguments that we might have passed.

This XML is parsed into the native operating system and language format and is then passed along to the target object. The object returns a response value to the SOAP server which then creates a SOAP response in XML as seen in Listing 4.

Listing 4. SOAP Response in XML

The response, like the request, uses HTTP and HTTP headers to pass some metadata, including the server type, date, content length, type ("text/xml") and even the type of SOAP server being run.

The envelope for this particular response, like the request, contains no header. However, it does contain a body, in which the return value (of type "xsd:string") is returned. While the request uses a namespace of "namesp3:capitalize", the response uses a namespace of "namesp1:capitalizeResponse". This is standard in SOAP; XML namespaces are used to identify whether the message contains a request or a response and for which method the response is being sent.

Without any explanation, Listing 5 is the similar debugging output from a call to capitalize_array(reuven, shira, atara):

Listing 5. Debugging Output

A SOAP Client

SOAPsh.pl can demonstrate interactive requests, but SOAP is much more useful when our programs can create and issue their own requests. Listing 6 demonstrates the code for a simple SOAP client that connects to our server on port 8080 of localhost. Notice how the URI is once again the name of the object, and the proxy is the name of the SOAP server.

Listing 6. soap-client.pl

What is particularly amazing about SOAP::Lite is how it allows us to invoke methods on our object that only exist across the network. That is, the “uri”, “proxy” and “result” methods obviously exist for the SOAP::Lite object. But the “capitalize” method only exists for our remote Text/Caps object. SOAP::Lite is normally smart enough to figure out the difference, passing along any method that cannot be locally resolved.

A CGI-based SOAP server

Our standalone server was meant to be simple, and it is. However, what happens when we begin to get millions of requests per day? Then our standalone server will no longer be able to keep up, and users' method calls will no longer be serviced.

Listing 7. cgi-soap.pl

A practical solution is to use a piece of software optimized for receiving many incoming HTTP transactions, namely Apache. Using Apache to handle our incoming SOAP transactions means that we can scale it as high or as low as we need. Our server program no longer needs to take this into consideration; it can focus on the mundane details of receiving SOAP packets and passing them along to Perl modules.

Creating a CGI-based SOAP proxy is not very different from creating a standalone program. Perhaps the most important thing to keep in mind is that you should not use CGI.pm or any other CGI-related module with which you might be familiar. Remember that the CGI program here is the SOAP proxy, and the CGI protocol is being used to transfer the XML-encoded request and response back and forth.

Other Goodies

SOAP::Lite comes with so many goodies, it's hard to know when to stop describing them. For example, those of you who have been convinced to use mod_perl in place of CGI will be pleased to know that SOAP::Lite has native mod_perl support, along with CGI and standalone support.

Your own programs can take advantage of the “autodispatch” mechanism we saw above, in which any method name not recognized locally is transmitted to a remote object.

SOAP::Lite can handle the transfer of most data structures supported by SOAP, including objects. In other words, you can invoke new( ) on a remote object, and then invoke various methods on the object returned by new( ). This functionality has existed for years on a number of specific platforms, but the fact that SOAP makes it platform-independent is truly amazing.

Finally, SOAP has grown beyond its exclusive use of HTTP and now supports a variety of other protocols, including some that we might not expect, such as POP3 and SMTP. SOAP::Lite supports all of these protocols; by the time you read this, it will undoubtedly support many more.

SOAP and Three-Tier Applications

Now that we have seen how SOAP can be used in simple circumstances, let's consider how it might be used in more complex situations. For example, let's assume that I have to create an extremely large web site that depends on a back-end relational database. In many cases, as regular readers of this column know, I will prefer to do the implementation in mod_perl and HTML::Mason.

But as the market for server-side Java grows, it's possible that some or all back-end functionality might be available using JavaBeans. Moreover, as Enterprise JavaBeans (EJB) becomes an increasingly pervasive and intriguing technology for distributed applications that require transactions, I might even prefer to do some of the implementation in Java.

With SOAP, I am now free to mix and match languages and platforms as I wish. If I create a SOAP server with access to the appropriate Java objects, there's no reason why my Mason components cannot communicate with a Java middleware layer. In some cases, this might even be preferable even if the system begins as a one-language affair. Given that Perl has no support for networked transactions, we might want to write an initial implementation in Perl and move toward EJB in the future. Using SOAP, this might be possible and even desirable.

Finally, web application servers are already beginning to work with SOAP. Not only does this allow objects on other computers and written in other languages to communicate with a given server, but it opens the door to Internet services that are not necessarily based on the Web. Perhaps web-based newspapers will begin to offer SOAP-based headline systems, taking the same content that is available on their web site but packaging it in such a way that someone can download a customized set of headlines with a single SOAP call. With such a service in place, users could install a desktop (non-web) application that would update itself to display the latest headlines every few minutes.

Conclusion

SOAP heralds the beginning of a new type of distributed Internet application, namely one that can perform remote procedure calls across operating systems and programming languages. No longer does RPC have to be a proprietary, difficult to-understand or difficult-to-invoke process; in the course of an afternoon, you can create a simple distributed application. Just what this means for the future of the Web and the Internet is a good question, but already some are claiming that desktop applications will increasingly be GUI shells that send SOAP requests to centralized servers. Regardless of what the future may bring, the fact that Perl and other free languages can use SOAP means that we will soon be able to communicate more easily than ever. And hey, isn't that the whole point of the Internet?

Resources

Reuven M. Lerner owns and runs a small consulting firm specializing in Web and Internet technologies. He and his wife Shira recently celebrated the birth of their daughter, Atara Margalit. You can reach him at reuven@lerner.co.il or on the ATF home page, http://www.lerner.co.il/atf/.

Not Cooking the Books…

Marcel Gagné

Issue #83, March 2001

Marcel gives HOWTOs on time tracking and bookkeeping with Linux for consultants.

Tu n'es pas sérieux, François! I admit you are a one in a million waiter, *mon ami*, but setting off to start your own consulting company is harder than you think. Since you are asking my advice, I will be happy to offer some help. But first, have you not noticed that our guests have arrived. *Vite François!* Show them to their tables.

Ah, bonjour, mes amis! Welcome once again to *Chez Marcel. François. Vite. Du vin* for our friends. The 1997 Vouvray Cuvee Constance should be wonderful. You see, *mes amis*, François was asking about what he would need in order to do a little consulting work on the side when he is not here at the restaurant. With his growing knowledge of Linux, he feels he might be ready to help others out and perhaps make a little *argent, non?*

Thank you, François! Please. Pour.

In many ways, Linux is the great friend of the computer consultant. Out of the box, your distribution contains a bevy of tools to get you connected in almost every way possible. Your Linux system is a mail server, a communications console, a Telnet client, an SSH client and so on. Programming tools of various stripes are there for you: C++ compilers, Perl, Python, Tcl/Tk and others. In fact, a typical Linux distribution has just about all the tools you need to become your own ISP or run a successful web site. What you don't necessarily have are the tools of business itself.

Oui, mes amis. It comes down to money. *Bien sur*, when you are out there working away, you are working because it will pay for your meals at home with the occasional night out at *Chez Marcel, non?* To get paid, you need to be able to tell your customers how much time you have spent working for them. Then, you must bill them, collect the money and keep track of the whole process. The life of the independent consultant is that of a self-contained, one-person business, and it should be treated as such.

Well, I am here to tell you that the same dedicated community of open-source programmers that have made Linux your most affordable, infinitely customizable workstation or server possible, have also been busy cooking up the tools for dealing with *la Banque*.

The first thing we will need to do is track our time. To do this, I am going to give you a couple of alternatives starting with a low-calorie command-line tool called **tt**:

```
tar -xzvf tt-1.0.tar.gz
cd tt-1.0
./configure
make
make install
```

To use tt, start by typing the command name on a line by itself. It will complain and tell you that it doesn't know what you want, but as part of that complaint, it will have generated a .tt directory under your home directory. Now, change to that directory and create a file called projects.conf. This is essentially a list of projects against which you wish to track time. Each project name is surrounded by square brackets. Here's a sample I was working on:

```
[cook] # Very important
[eat] # You can make money doing this?
[drink_wine] # If only it were possible
[write_recipes] # The real work, non?
```

You can, at any time, list all your current projects with the **tt '*' -list** command. To start tracking time against a project, I would type the command name with a **-start** flag. For instance, drinking wine, while pleasurable, is part of the job and something that I must *regretfully* charge for. (I joke, of course.) But if it could be so, I would start billing for it like this:

```
tt -start drink_wine
```

When I am done with my, ahem, work, I would issue the same command with the **-stop** flag.

For a simple program, tt is quite powerful. One interesting use of the program is as a time wrapper for tasks. Say you are working for Henri's Fine Wines. You are doing work on the company web server setting up his e-commerce site. The work is all done remotely, and you Telnet into the server to do your work. You could wrap the Telnet command with the tt command to track your time automatically. Like this:

```
#!/bin/bash
tt henri_wines -start
telnet henri_website.com
tt henri_wines -stop
```

C'est bon, non? When it comes time to report, you can generate a quick report of your time with this command:

```
tt drink_wine -export
project 'drink_wine' (closed):
  Tue Dec 19 2000 16:53:31-Tue Dec 19 2000 16:53:56: 00h00m25s
  Wed Dec 20 2000 12:10:39-Fri Dec 22 2000 16:52:59: 52h42m20s
  Fri Dec 22 2000 16:56:18-Fri Dec 22 2000 17:01:04: 00h04m46s
+ Tue Dec 19 2000 16:53:31-Fri Dec 22 2000 17:01:04: 52h47m31s
```

Even better, you can use the **-format** modifier to, export to and even create, an SQL database. tt knows about MySQL and PostgreSQL. For instance, I could create a database in PostgreSQL with this command:

```
createdb myconsultancy
```

Next, I export my tt data with this command:

```
tt drink_wine -export -format pgsql | psql myconsultancy
```

With that information in place, I could generate custom reports from my PostgreSQL database:

```
psql -c "select * from tt_timing;" my_consultancy
```

You may find that tt does everything you want it to, which would be wonderful, but if you are on the road a great deal and you need some way to file your time remotely, then this next item on the menu might be just what your taste buds demand. You may also have become highly successful in your consulting career and have hired many people to help you out. Again, this may be just what you are looking for.

One of the great things about browser-based applications is that they allow you to do your work from nearly anywhere. Start up your browser, set the URL for the appropriate link, allow a few seconds for the link to rise to the display window, and *Voilà!* You have a fresh, steaming application ready for use. In the world of time tracking and billing, I was fortunate enough to sample onShore Inc.'s “TimeSheet” program. This is a great application and worth checking out. It offers a clean, simple interface, a separate administrator's screen as well as profiles for both full-time employees and contract consultants. You can also generate reports based on the individual consultant or a specific client.

```
tar -xzvf onshore-timesheet-current.tar.gz
cd onshore-timesheet-2.2
```

Before you go running off excitedly typing “make”, I must tell you that your humble chef failed to take the first rule into consideration when trying this application. That rule is “Thou shalt read any and all INSTALL and README files”. After spending some time with a delightful little Merlot, I once again visited OnShore's TimeSheet and discovered that a little editing of the Makefile was all I needed to do to make things work. The changes required are few and everything is near the top of the Makefile, but I will highlight a few:

```
PERLINC         := /usr/lib/perl5/site_perl/5.005
APPROOT         := /usr/local/apache/htdocs/timesheet
TIMESHEET_URL   := /timesheet
CONFFILE        := timesheet.conf
GUEST_ACCOUNT   := no
WWWUSER         := www
APPOWNER        := www
```

Since onShore TimeSheet comes with a couple of necessary Perl modules, you should modify the PERLINC parameter to reflect your **site_perl** install directory. The one above is mine. The **APPROOT** represents the path to the application on your web server. The default is Red Hat's installation default of **/home/httpd/html**. Change it to reflect your own setup. My web servers are all rebuilt Apache servers with a /usr/local/apache/htdocs document root.

The next change is the default URL for the application. If you choose an APPROOT with “timesheet” as the eventual directory (the default), you will want this to be **/timesheet**. Next, decide on a name for the configuration file. I chose timesheet.conf. The install will create a default “guest” account if you so wish. I decided that I would create all my users as needed. The last two parameters are both set to WWW for me. The WWWUSER is the default user id your web server runs as. On many systems, this is set to the user “nobody” by default. You should decide accordingly. Finally, you want to set the APPOWNER. This sets the ownership for the whole application.

Now, your patience will have paid off. If you are running on a Debian system, type **make install-debian** at this point. Otherwise, type **make install**. You will see some exciting information scroll by on your screen as a default PostgreSQL database is created for you along with your administrator user ID. You are almost there. To run this, you need to make sure that your web server understands .cgi extensions and executes them appropriately. This is done by adding the lines below to your **httpd.conf** file. Note: Depending on the *vintage* of your web server installation, the appropriate configuration file might actually be **access.conf**. The two most likely configuration directories are **/usr/local/apache/conf** and **/etc/httpd/conf**.

```
AddHandler cgi-script .cgi
<directory /usr/local/apache/htdocs/timesheet>
Options +ExecCGI
AddHandler cgi-script .cgi
</directory>
```

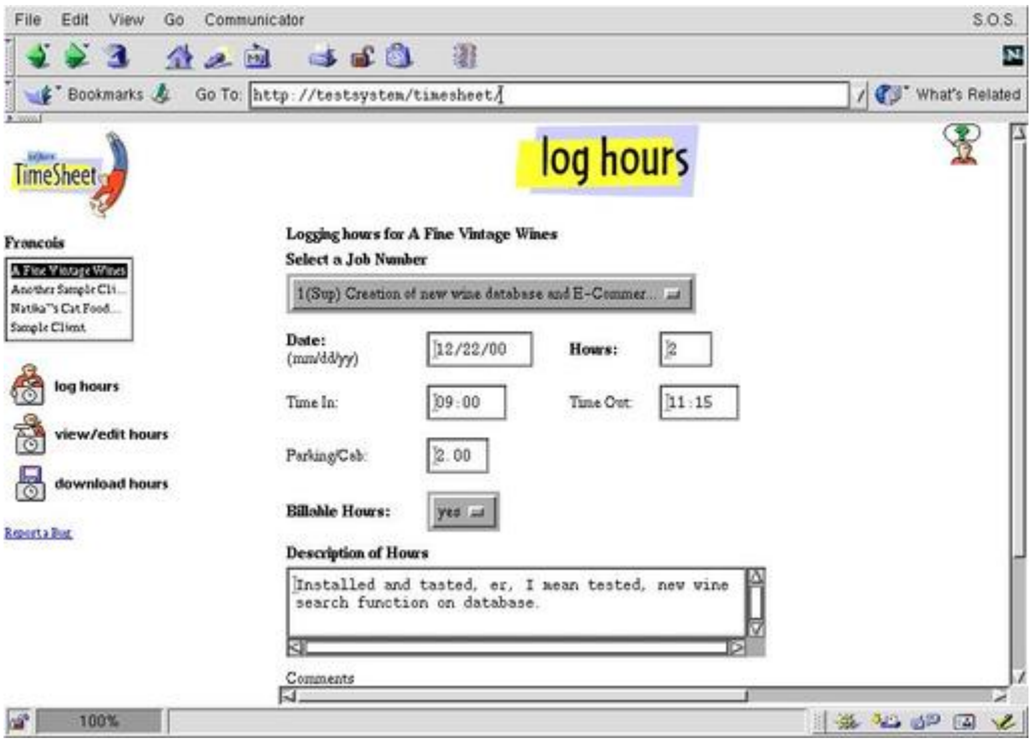
The **<directory>** paragraph is there so you can execute cgi files wherever you decided the installation is going to be. Remember that cgi scripts are normally executed from cgi-bin rather than html or htdocs.

The last thing you need to do is make sure that PostgreSQL is started with the -i option. For instance, in my /etc/rc.d/init.d/postgresql startup file, (Debian users should check /etc/init.d/postgresql instead) I have the following line:

```
su -l postgres -c '/usr/bin/postmaster -S
-D/var/lib/pgsql'
```

I simply changed the default start so that there was a -i option included after the word “postmaster”. After restarting PostgreSQL and your web server, you are ready to roll. Simply point your web browser to the TimeSheet URL: http://my_webserver/timesheet/.

You'll need to give your admin login name and password, which is “admin” for the login and “admin” for the password. Obviously the first order of business should be to change this to something more secure. This is also the time when you want to start creating your users, be they permanent employees, consultants or other administrators. Check out Figure 1 for a look at the onShore TimeSheet program in action.



Go To: http://testsystem/timesheet/

Logging hours for A Fine Vintage Wines

Select a Job Number: 1(Sup) Creation of new wine database and E-Commer...

Date: (mm/dd/yy) 12/22/00 Hours: 2

Time In: 09:00 Time Out: 11:15

Parking/Cab: 2.00

Billable Hours: yes

Description of Hours: Installed and tasted, er, I mean tested, new wine search function on database.

Comments

Figure 1. Timesheet Screenshot

Magnifique! You can now keep track of your time. From your detailed timesheets, you can invoice customers and reap the fruits of your labours, *non?* Collect the money and let your accountant worry about the rest. *Une minute!*

The accountant will want some kind of financial records. This brings us to the great dreaded beast of any business enterprise: keeping a set of books. Fortunately for the Linux user, there are answers here as well. Come, *mes amis*, and experience a little sample of accounting packages for Linux.

Accounting applications in the Linux world represent some of the most requested of tools. Luckily, these are starting to appear in greater numbers. These may be commercial applications, GPL'ed, or free software. Since we are discussing open-source applications, we will concentrate our menu items in this area. If your application is not too complex, you may want to start by having a look at **BANAL**, which stands for BANAL's Absolutely Not A Ledger. Written by Matthew Rice, BANAL is a fairly simple application with big dreams. I say that because BANAL can do quite a number of things, even if it does them simply. With BANAL, you can track your time, maintain clients and suppliers, invoice, track expenses, write cheques and more. *Et oui*, BANAL will cost you nothing but some time.

The software is available as a free download with a BSD-style license and is fairly easily installed. I should point out that while the source code is freely available—actually, it is all source since this is a Perl application—and that you can get it as a tarball, your chef humbly recommends getting the RPMs instead. There are two RPMs: a client and a server. The source tarball contains a couple of extras such as some evolving PHP scripts. That may be worth the extra work to some. The RPM install is quick and easy:

```
rpm -ivh banal-*
```

Quite a bit happens here. Among other things BANAL creates a user in your /etc/passwd file and starts the server. BANAL is really quite simple to use, but I still recommend that you read the quickstart file that comes with the install. You'll find it in **/usr/doc/banal-client-0.10/quickstart.txt**.

Before we move on, I should give you a little bit of a warning. On my system (and perhaps on yours), the **bk** program tried to open /usr/bin/vi by default, whereas on my system the path to **vi** was /bin/vi. You may want to create a symbolic link to rectify that problem or simply assign the EDITOR environment variable, which will override BANAL's built-in path to the editor. This also gives you the opportunity to choose the editor you prefer:

```
export EDITOR=pico
```

Once this is all done, you need to start creating your accounts, profiles, suppliers, expense accounts and so on. This is done using the one command **bk**.

```
BANAL-FTL bookkeeping system
 bk is BANAL's client tool for the command line. Try:
 bk help commands       for a list of commands
 bk help command        for help on a specific command
 bk help usage          for generic command-line arguments
```

Getting started requires that you create one initial client (your company) using the command **bk client your_co_name**. Make sure you fill in all the appropriate or required fields, then save your work. Next, you need to change the company information settings to show that this client profile is actually your company:

```
$ bk setting /UserInfo/CompanyID
```

You'll find yourself in editor mode with something resembling the following. (Remember to change the "Value" field to something other than "ChezMarcel", *non*?)

```
# A BANAL Setting Specification.
#
# Setting:     Unique Setting ID
# Value:       Value                  @@ REQUIRED
# Description: Item description
Setting:     /UserInfo/CompanyID
Value:       ChezMarcel
```

Ah, it is all so easy, *non?* As with the timesheet part of your consulting business, there is a graphical alternative here as well that goes by the name of **GnuCash** [look for an article on GnuCash in *LJ*'s April 2001 issue]. This is a wonderful open source, GPL'ed and completely free accounting package.

I decided to build GnuCash from scratch. To start, I downloaded the source for the latest package version from the GnuCash web site. For those who are a little more impatient, you will find some precompiled binaries at the site.

The core of this application uses the GNOME libraries, so you will need to have these loaded regardless of whether you choose to build from scratch or use the binaries. Obviously, those of you who are already using GNOME as your desktop will find that gnome-libs is already there. For those who might be using another desktop, you will need to load the gnome-libs and the supporting libraries, specifically, GTK. You'll find that GnuCash is worth the work, but it does require a bit of hunting and gathering. In addition to the GNOME framework, you also need **guile**, **swig**, **slib** and **g-wrap**. The final package, g-wrap, is actually on the GnuCash FTP site's download area. There are a few other requirements, most notably **libjpeg** and **libpng**, but these will likely already be on your system. The slib package may already be there as well in the guise of the **umb-scheme** package. Check the Resources section for pointers to all these sites.

The first step, after obtaining the software *bien sur*, is to extract the archive. This is followed by some very familiar steps:

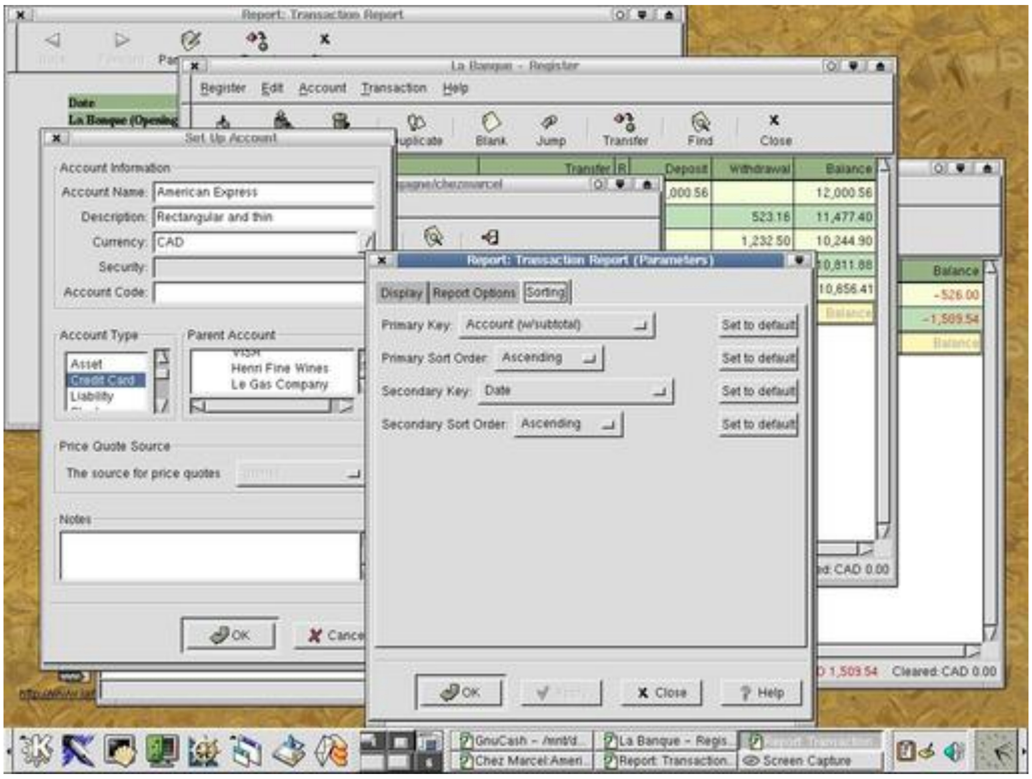
```
tar -xzvf gnucash-1.4.9.tar.gz
cd gnucash-1.4.9
./configure
make
make install
```

At this point, you should probably check out a couple of fairly important links. When your humble chef first tried this recipe, things did not go altogether smoothly, specifically as related to slib. The guile libraries make some assumptions as to where the slib information will wind up located. This is apparently not a problem on all systems, but it was on my Red Hat system. I changed directory to /usr/share/guile and created the following symbolic links:

```
ln -s /usr/lib/umb-scheme/slib slib
ln -s /usr/lib/umb-scheme/slibcat slibcat
ln -s /usr/lib/umb-scheme/slib/mklibcat.scm mklibcat
```

The umb-scheme package is my Red Hat installed-scheme libraries (remember slib earlier). With these links in place, things went a lot smoother. Once this is done, start GnuCash by typing **gnucash** at the command prompt.

If you have ever used Intuit's Quickbooks package, you will find GnuCash invaluable. With its easy-to-use interface, you can create your chart of accounts for banking, credit card transactions and so on (see Figure 2). The package has a nice reconciliation feature, and numerous reports including a profit and loss statement so you can see just how much money you have made at any given time. It even handles multiple currencies for you globe-trotting consultant types. The only downside that I can think of for a small business is that there is no invoicing function. That, you will have to do manually.


Report: Transaction Report
La Banque - Register
Register Edit Account Transaction Help
Duplicate Blank Jump Transfer Find Close
Set Up Account
Report: Transaction Report (Parameters)
Display Report Options Sorting
Set to default
OK Cancel Close Help

Account Information
Account Name: American Express
Description: Rectangular and thin
Currency: CAD
Security:
Account Code:
Account Type: Asset, Credit Card, Liability
Parent Account: Henri Fine Wines, Le Gas Company

Primary Key: Account (w/subtotal)
Primary Sort Order: Ascending
Secondary Key: Date
Secondary Sort Order: Ascending

Figure 2. GnuCash Screenshots

Mon Dieu, François! It is late and there is still much to do. Pour our friends another glass. What is it, François? You have decided to abandon the idea of doing extra work as a consultant? *A cause de moi? Mais non*, I do not give you too much to do. You have plenty of time! We will discuss this later, François. For now, please tend to our guests. More wine.

Mes amis, if you do decide to go it alone, always remember that, truly, you are not alone. Having chosen Linux, you have at your disposal all the tools needed to begin and operate your high-tech consultancy. With a little help from some great open-source chefs, you may even find that your own ideas will join theirs, and someday your own recipes may be featured here at *Chez Marcel*. Until next time, your table will be waiting.

À 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 edits TransVersions, a science fiction, fantasy and horror anthology. He loves Linux and all flavors of UNIX and will even admit it in public. In fact, he is currently working on *Linux System Administration: A User's Guide*, coming soon from Addison Wesley Longman. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things from his web site at http://www.salmar.com/marcel/.

Debian Multiboot Installation

Robin Rowe

Issue #83, March 2001

Let's build a Linux multimedia workstation that will boot Linux, Win98, Win98SE, WinNT or Win2K from LILO.

In this month's column we will walk through the steps of installing Debian Potato in a multiboot configuration with Windows 98/NT/2K. Our machine will serve as the desktop system of a software developer (me) making the transition from supporting only the creation of Windows programs to supporting Linux. Before we're done we will discover that several things in the documentation can be misleading, and it's not advisable to follow blindly the Debian instructions or Linux HOWTO documents that you find on the Web.

Our machine, primarily used as a Windows software-development system, is an Abit BH6 motherboard Intel Celeron 300MHz PC with 128MB RAM, Yamaha CDRW, SoundBlaster Live!, Haupaugge WinTV, Hercules Terminator 2x/i 8MB and two 20GB drives. That may not seem like a very fancy machine, but it meets our needs. Our philosophy is to avoid getting too far ahead of the curve with development machines. We want to develop software on systems not too unlike those our users have. This helps avoid unpleasant surprises such as inadvertently creating software that won't run properly on anything but the newest and most powerful PCs.

To avoid risk to the data on our existing drive, and also because we're out of space, we have just installed a new 20GB Maxtor DiamondMax Plus EIDE Ultra ADA/66 drive. A 7,200RPM drive, the Maxtor is faster than our old 5,400RPM 20GB drive. That speed is needed for doing work with video that tends to get I/O bound. It's amazing how cheap big fast drives have become ($130 US). To install the drive we had to drop the power supply and pull the AGP card to get into the bay. The drive came with a Caldera DR-DOS boot floppy containing documents, partition software and drive image copy software, but we didn't use any of that. We swapped the ribbon cable from our existing drive (temporarily disabling it) to our new drive, booted to BIOS, did auto-detect of the drive, took the default LBA setting and away we went. For all intents and purposes, we are doing our install to a bare system. Our old drive is deliberately off-line as a safety precaution.

Let's start, strangely enough, with installing Windows. Versions of that OS aren't particularly friendly to any other operating systems already residing on a PC. We can avoid some hassle by installing Windows first, then Linux afterward. Windows 98SE sells for about $195 US and Win2K for about $265 US. Since we build our PCs from parts, we don't get Windows included with our PCs.

Windows 98 has three major versions: original, Second Edition and Millennium Edition. We prefer SE or ME because support for multiheaded monitors and other nifty device features was added. We install Win98SE by booting from the CD (twice), choose non-large disk support (FAT16), create a 2GB partition, provide int 3 and 0x300 parameters for our non-PNP ISA network card and enter our behind-the-firewall IP of 192.168.1.2 with netmask 255.255.255.0. We use the same login name and password for Win98SE and Win2K (not administrator) because that means less hassle when opening network Windows drives on a LAN.

Our network gateway is WinGate running on a WinNT box on our LAN at 192.168.1.1. This seems totally backward perhaps, because Linux is recognized as a better firewall than WinNT, but that's our setup since we are starting from a Windows-only shop. This turns the notion inside-out that Linux must be the server and Windows the desktop, a misplaced idea that may hold typical Windows users back from trying Linux on the desktop.

We boot Win98SE successfully from the hard drive and connect across our LAN to our treasure trove of downloaded Win98 drivers. Having a second PC available while building a PC can speed up the process considerably. One of the first things to do is install the correct video driver so that we can get out of 640x480 VGA mode. Here we encountered our first trouble. Win98SE has a redesigned video subsystem that chokes on the Win98 driver for our Hercules Terminator 2x/i video card. The screen goes blank. We need to boot in safe mode (plain VGA), a task made more challenging because the magic keystroke to boot safe mode in Win98SE is Ctrl and not F8 as it was in Win98. We couldn't figure this out at first. Windows doesn't display what keystrokes are available at boot up. Cutting the power forces a safe mode boot. We install an alternate driver from WinDrivers.com. The rest of the Win98 install goes without incident.

We installed Windows 98 first because it is the least tolerant of other operating systems. It has a proclivity to overwrite the MBR (Master Boot Record). Next we install Windows 2000 Professional. The pro edition is the desktop version, the replacement for Windows NT Workstation. There is no amateur version; the other flavor is Server. We boot Win2K from CD and create a 2GB partition in NTFS. The Win2K install goes flawlessly. We can now boot Win98SE or Win2K using the Windows boot loader c:/ntldr. We are careful to create emergency disks for both Win98SE and Win2K. Fortunately, we won't need them.

We created our Win98SE partition as FAT16 because Linux and MS-DOS can read/write FAT16. There isn't any benefit to FAT32 unless the drive partition is larger than 2GB (if you at some time use Win98 fdisk, refuse large disk access). We want to use NTFS with Win2K for many reasons, but mainly because it avoids the time-consuming Windows scandisk process if we have a crash. A hard reboot on NTFS is no fuss normally. We prefer to put an OS on its own 2GB partition because we have found that increases reliability under Windows. Now, let's install Debian Linux.

According to Debian founder Ian Murdock, the most unique aspect of Debian development as compared to other Linux distributions is the fact that it has been and continues to be developed openly by a group of volunteers, and that it is open to other volunteers who wish to join the effort. Debian is not developed by one individual or a small, closed group. Instead, it follows in the tradition of the Linux kernel; it is developed by those who use it, and this makes for a higher quality, more dynamic and truly modular system [see *LJ* November 1999, “Overview of the Debian GNU/Linux System”].

The Debian web site (http://www.debian.org/) points out that Debian GNU/ Linux provides more than 3,950 packages of precompiled software bundled for easy installation. It should be noted that Debian is widely recognized as the most pure distribution in the free software and open-source philosophy. Free software advocates will enjoy reading the Debian “social contract” at http://www.debian.org/social_contract/. With Debian there is no company in the sense that there is for Red Hat. That's not to say you can't get commercial support from a company like LinuxCare if you want it. But, it is as a community of mutual support that Debian especially appeals to Linux experts and programmers, and the community support available on the Debian Users' list is awesome. We found that we typically receive the correct answer to a question there within ten minutes. And, the tone of the group is a joy. If you ask how to do X, you get how to do X, not that you should do Y, or a question asking why you want to do X—simple, to the point.

Debian releases have names. The current release 2.2 is called Potato, and the next release will be Woody. Initially we tried to download Potato by FTP from a mirror site but found the instructions too confusing and the download too time consuming (on 200KBs cable-modem). Many vendors offer Debian on CD, and some offerings have more CDs than others. We got the full six-CD set from Linux System Labs (http://www.lsl.com/) for $9. This was unbelievably cheap compared to what Windows costs. When ordering you can also “buy” a $5 contribution to the Debian organization if you like.

Installing Debian your first time is, to put it bluntly, sys admin hell. The documentation tends to steer you away from what a mere mortal needs to create a successful install. A forest of good advice, bad advice, conflicting opinion and outdated instructions—the documentation doesn't need to be that hard. And in hindsight, maybe it isn't that hard. It's just very confusing for first-timers. Back when I taught computer science at the Naval Postgraduate School in Monterey I was exposed to a term I hadn't heard before (or since). Naval aviator students would ask for a gouge. That is a document that accurately gives all the important information but only the relevant information. Gouges are intended for capable newbies and not written with painful detail. Debian could really use some gouges. We'll guide you around the confusion here.

Our first Debian problem: the LSL Potato CD won't boot. For whatever reason it doesn't seem to have been made as a bootable CD. We try to start it from Windows using **E:/install/boot.bat**, but the documentation points out that the command only works in MS-DOS mode, not the emulated DOS box console. In Win98SE, we pick “Restart in MS-DOS mode” from the shutdown menu. That boots us into true MS-DOS just fine, but do we have the necessary real mode DOS drivers to be able to read a CD? No, of course not! Time to stop and rethink.

The on-line “Debian GNU/Linux: Guide to Installation and Usage” document explains how to create boot floppies. The instructions are wrong because the file names and CD layout have changed for Potato, but the basic theory is correct. We ignore numerous warnings we find on the Web that boot floppies are evil. We have no trouble.

Here's how to boot Potato from two floppies when you can't access the CD from DOS:

1. In Win98SE copy the Debian CD directory of E:/install to C: on your hard drive.
2. Restart in MS-DOS mode. The next step, making floppies, won't work at the Windows DOS console.
3. At a real DOS prompt execute C:/install/rawrite2.exe. (If you didn't follow our advice to make the partition FAT16 instead of FAT32 you are hosed.) Enter **rescue.bin** and **a:** at the prompts.
4. Repeat with root.bin and a:.
5. Boot using the created rescue diskette, and *voilà*, you have a minimal Linux running that can see the CD.

Partitioning your hard drive can be scary. There's something about the danger of irrevocably losing all your data. The Debian installer will put you into Linux **cfdisk** which works very nicely but is intimidating at first. Standard warning: to avoid problems only partition using software intended for the OS you are partitioning for.

Something to keep in mind with partitioning is that there is only room for four entries in a drive's partition table. Primary or extended partitions use one entry. However, logical partitions within a single extended partition don't count. Win98SE is a primary partition. WinNT or Win2K are created by default as logical partitions. With any OS that will let you, and for all data partitions, use logical partitions to avoid running out of partition table entries. If you do run out you can't partition whatever space is left on the disk. Linux needs two partitions: a boot and a swap partition. We decided to make both Linux partitions together a total of 2GB, subtracting 250MB for swap. We should have put Linux on a logical partition, but not knowing any better we made it a primary partition. This needlessly wasted a slot in the partition table.

The Linux LILO boot loader is another program that strikes fear into new users. Despite the reputation, its installation was easy. We are glad we didn't follow the widespread advice of making Windows **ntldr** our primary boot loader (as suggested in linuxdoc.org “Win95 + WinNT + Linux Multiboot Using LILO Mini-HOWTO”). Having LILO boot everything is more elegant and was less trouble to install, too. When prompted, install LILO in the MBR (the default). This will temporarily disable Windows booting. The next step is to regain access to ntldr from LILO.

You touch up /etc/lilo.conf to point to Windows as an “other” boot OS using the syntax as documented for multiboot into DOS. By the way, if you don't already know how to operate a Unix text editor such as vi you will definitely learn. Don't forget to touch up /boot/bootmess.txt and run **/sbin/lilo** to take all your changes. Reboot and LILO puts us into ntldr (when asked) where we see the usual Windows OS boot selector screen. Win98SE boots now. We have jumpered our old drive as slave device, connected it to the same ribbon cable as our new drive, then booted and auto-detected the drives in BIOS. Because we have a second ntldr on /hdb we do some extra magic using the little known map directive in lilo.conf to enable swapping boot drives. Now we can boot Linux, our new copy of Win98SE and our old copies of Win98 and WinNT:

```
other=/dev/hda1
        label=Win2k
        alias=2
        table=/dev/hda
other=/dev/hdb1
        label=WinNT
        alias=4
        table=/dev/hdb
        map-drive=0x80
```

```
           to = 0x81
           map-drive=0x81
           to = 0x80
```

Win98SE boots fine, but we get an “ntoskernel missing” error trying to boot Win2K. The answer is to bump up the Windows partition number in c:/boot.ini. When a primary partition is installed later (as we did needlessly with Linux) it can bump up the logical partition numbers. Just add one to the partition number. Logical partitions must be contiguous to be in the same extended partition:

```
multi(0)disk(0)rdisk(0)partition(3)\WINNT="Microsoft
    Windows
2000 Professional" /fastdetect
```

We set the Linux network interface settings:

```
# /etc/network/interfaces -- configuration file for ifup(8),
# ifdown(8)
iface lo inet loopback
iface eth0 inet static
        address 192.168.1.2
        netmask 255.255.255.0
        gateway 192.168.1.1
```

We point to our firewall DNS in /etc/resolv.conf:

```
nameserver 192.168.1.
search 192.168.1.1
```

To recap, the sequence we followed was to install all Windows operating systems first, then Linux. Heady with success, we can now boot Linux, Win2K, Win98SE, WinNT or Win98 using LILO! Next month we will configure XFree86 so we can start X Window System and then patch kernel 2.2.17 to install Video4Linux. That will give us television video on our Linux screen.

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*, *Data Based Advisor* and has had many papers published in conference proceedings. 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/, and an automated television news monitoring system developed for DARPA and the Pentagon. He has taught C++ at two universities and designed video software in Fortune 500, DoD and academic environments. You can reach him at robin.rowe@movieeditor.com.

Return of the Bazaar

Doc Searls

Issue #83, March 2001

Doc discusses the evidence of a widespread reacquaintance with reality.

You shall no longer take things at second or third hand...nor look through the eyes of the dead,nor feed on the spectres in books.You shall not look through my eyes either,nor take things from me.You shall listen to all sides and filter them for yourself. —Walt Whitman

I sense an era ending.

Of course that's an easy call when the economy is falling like a bad tent, but there's something more subtle and serious than the Law of Cycles at work here. I think what's happening is a widespread reacquaintance with reality—the reciprocal of disillusionment. It shows up in what we're not buying: needlessly faster computers, humorless comedies on TV, specious investment schemes, vacuous campaign promises and promotional malarkey in all its forms.

This seems to go beyond skepticism, mass ennui and declines in disposable cash, though all those factors play. It comes from sane judgment—the kind that rises from conversations with people we trust, rather than from herd mentalities or propaganda machines that try to command and control what we think, eat, buy and love. More significantly, it comes from our own sense of self-worth.

Around two years ago, I was involved in just such a conversation. There was a growing sense among the four participants that together we knew something that each of us alone couldn't quite sum up. Finally, after a couple months of back-and-forth, one of us (Christopher Locke, aka RageBoy) put what we all knew into a few simple words: “We are not seats or eyeballs or end users or consumers, and our reach exceeds your grasp. Deal with it.”

Soon as we heard it, we knew we had to say it again, louder and in more detail, because we were speaking for the billions of other people who also sensed the same thing but hadn't started talking about it yet. The result was *The Cluetrain Manifesto*.

More than a few people (occasionally including myself) have thought that *Cluetrain* was mostly about the shift of market power from supply to demand. While that's true, I think *Cluetrain* was also about returning to the true meaning of the word market, which finds its best synonym in *bazaar*--a place where people come to do business and make culture.

Markets were around for thousands of years before we started using the word to label product categories, geographies, demographic groups and demand itself. All those uses awaited the rise of industry, which began about two hundred years ago. Throughout the Industrial Age, talk about markets has been mostly a supply-side activity. When we hear consumer electronics, office supplies, Mexico and BMW drivers all referred to as “markets”, we hear the supply side talking to itself. To the demand side, the market is still mostly a place to shop—a bazaar.

Nothing has done more to raise bazaar consciousness than the Open Source movement, starting with Eric S. Raymond's *The Cathedral and the Bazaar*. While Eric didn't have much to say about markets per se in *The Cathedral and the Bazaar*, he did much to popularize the bazaar as a conceptual metaphor. Google counts 355,000 web pages that contain the word bazaar, and I would bet at least half of them wouldn't be there if Eric hadn't spread the meme. And what better conceptual metaphor for market than its most literal synonym?

So now the Open Source movement finds itself with two jobs. One is explaining itself, and the other is explaining what markets are really about. We already know the first isn't easy. Neither is the second.

For generations we've been looking at both business and markets from the industrial producer's position—one so corrupted by the abstraction of demand that it can barely comprehend the face-to-face, keyboard-to-keyboard nature of life in the bazaars that markets naturally long to be. What we call “consumerism” is really producerism, a tired ideology that believes it can forever categorize and organize markets for its own convenience.

For a case in point, try to buy a computer from a mass producer. If you're going to the biggest sources—Dell or Gateway—be prepared to declare your category. No, not “server”, “desktop” or “laptop”. We're talking about your category here. What are you? they ask. Are you a consumer, a business or something public like a local, state or federal government? Those are the cattle chutes through which these producers want customers to click their way toward a purchase.

That's rather annoying if you think of yourself more as connoisseur than category. But who are you? More accurately, what are you? To mass-producers, you're a consumer. Your job is to consume. Or, in the perfect words of Jerry Michalski, you are a “gullet”. That is, a creature “who lives only to gulp products and crap cash”.

Millions of dollars move through Dell's cash plumbing every day and not much less through Gateway's. Near as I can tell, there isn't much negative burpage on the gullet side of the two companies' transaction mills. Both enjoy wide regard as exemplary producers. Their products tend to show up at or near the tops of ranked reviews. So does their customer support. This begs a fair question: what's the problem here?

Maybe, for Dell and Gateway, there isn't one. But I'm not so sure. I think Dell and Gateway are just stuck with default mass-marketing metaphors they'd rather do without if they knew how customer-hostile those metaphors really are.

I was looking for data on the matter a few weeks ago when I visited my new local Gateway Country Store. Wearing my invisible objective journalist hat, I tried to be as positive and open-minded as possible. It wasn't easy. The first thing I noticed after passing through the door was a display on a kiosk that was brain-locked into an error message that nobody had bothered to correct, even though sales people seemed to outnumber the customers. While I'm sure the electronic corpse on the kiosk had been dead for hours, mere seconds passed before I was engaged by a sales guy. A surreal conversation followed.

“What do you have right now?”

“Well, we run a home office with a bunch of Linux and Mac boxes,” I said. “We've got DSL running to the house, and the boxes are all hubbed through Ethernet and a router to the Net. I'm thinking of getting an all-purpose Windows box, running the same software my accountant uses, testing out some multimedia stuff...”

“Then you want Windows 98.”

“Really? 98? Not Millennium Edition or 2000?”

“You can't run games or multimedia on Windows 2000.”

“Really? Why not?”

“You can't use sound on a network.”

“Huh?”

“Windows 2000 is a network OS. It doesn't do sound, except with drivers we don't support. And if you're running a network you don't want any of your employees downloading and playing MP3s at work.”

“I do that all the time.”

“If you're running games or multimedia you're doing consumer stuff. You want the consumer OS. If you want to set up a business network, you need to talk to somebody on our business staff.”

Eventually I left without buying anything. Meanwhile I observed that the place didn't seem very busy, even though this was the height of the Christmas season. I also noticed that Gateway was trying its best to do consumer marketing here. In other words, herding consumers into corrals where they could be further sorted into one sales chute or another. Nothing new, of course, except that it felt real old.

Yeah, we're in an economic malaise right now. But I've also been to Fry's and Best Buy, and those guys didn't seem to be hurting for customers. They also don't seem to be going out of their way to make customers declare their categories. Maybe that's because they're real stores, not just channel strategies, which means they're about markets, not just marketing. In real markets—bazaars—customers have choices. The more, the better. That's what we like most about Fry's and Best Buy, especially since prices for the same products are in the same low range.

Choice matters on the supply side as well, and it doesn't just take the form of the plethora. Those of us who operate with the most autonomy and honesty on the supply side do so by choice. We seek out the best work of our peers, take an interest in it, borrow from it, talk about it, join in on some work or go off and do our own thing. In traditional market terms, what we practice is our craft. There is maximum choice on our side as well as the customer's. That's how a bazaar—a real market—works.

Can big industrial producers have crafts? I think they can, as long as they maximize the choices on both the supply and the demand sides of their markets. That means they have to be open. They have to expose the source of what makes them worthwhile as a company in the world: namely, their own employees.

“In the past, people were cheap and technology was expensive,” Don Marti says. “Now it's the other way around.” In that case, producers need to expose exactly what makes the company most valuable in the marketplace, and then trust the market to do what markets do best.

Maybe then they'll start listening to what we've been saying out here in the software bazaar. And we can listen to them, too.

In fact, it's already happening.

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

Maragda: Running Linux from CD

Jordi Bataller

Issue #83, March 2001

An overview of one man's DIY adventure in putting Linux on a bootable CD-ROM.

The project described here is further proof that Linux is a powerful and versatile system.

My college has several trolleys that carry a computer (PC) and a VCR, and both are connected to a video projector. They have so many users that it's difficult to keep them working properly. This is especially true in the case of the computer; you can't ever be sure what software you will find on the hard disk.

I decided to use the projectors in the introductory programming course to show my students source C programs and how to run them step-by-step. I, of course, wanted to use Linux and its programming tools so that my students learn that Windows is not the only choice.

In this situation, I came to the conclusion that I needed some sort of portable Linux system, and I didn't want to use my notebook in the classroom.

My first idea was to try installing Linux on an Iomega Zip 250MB drive for parallel ports. After a few days of work I achieved it. It needs a floppy to boot the system, and the run speed is not too bad. Then I repeated the task, this time installing Linux on an IDE hard disk contained in the IDE box adapter for parallel ports. This configuration removes the limitation of 250MB disk space and speeds up the running of Linux.

From the lessons learned from these tasks, I asked myself, why not run Linux from a CD-ROM? That would be convenient and easier to use. In addition, students could use it to start learning Linux without having to install it on their computers. And so Maragda was born.

How to Use Maragda

The final product is a bootable CD-ROM, so you must configure your BIOS to boot from the CD drive. If your BIOS doesn't support this, there is a solution: a bootable floppy that will also boot Linux from the CD. In fact, a raw image of the floppy is what the CD contains to boot itself.

The Linux software on the CD is installed from the Red Hat 6.2 distribution, and I prepared two arrangements. The first one is a full installation. It contains the base system, printer support, the X Window System, the VGA16 and framebuffer X servers, GNOME (or you can edit .xinitrc and uncomment the line for the window manager you like the most), network server workstation, authoring and publishing tools (LaTeX, etc.), **Emacs**, development tools (**make**, **egcs**, etc.), DOS/Windows connectivity, mail, WWW and news tools, and other packages such as **rhide**, ssh support and the JDK 1.2.

The second arrangement, which I call Maragda Teaching, is only equipped with part of the packages of the base system and X (to use **fvwm2**), with the complete development tools and other tools such as **gv** and rhide.

Full Maragda requires at least 64MB of RAM, while Maragda Teaching (and its tools) runs with only 32MB. There's a second arrangement of full Maragda that runs with 32MB but does not include room for tools such as Emacs or Netscape, which require more memory.

In both cases, the same kernel is used. It is version 2.2.14, and it supports (among other things) the network, the framebuffer device, the loop-back device, RAM disk and initial RAM disk.

Configuration

One of the problems of any system intended for portability is how to adapt to different hardware. At the very least, the mouse and video (X) must be configured, and then the passwords, network identity and others. In the case of Maragda, it should be remembered that no change survives one session because the files on the CD are read-only.

Full Maragda is configured by default to support a PS2 mouse and the framebuffer device for video with a depth of 32. Network uses the driver 3c59x (a 3Com card) with identity maragda.gnd.upv.es (192.168.0.1).

However, full Maragda has an open door that allows you to decide on the configuration yourself. Just after the main file systems are mounted, the boot process stops and asks for an ext2 formatted floppy. Every file on the floppy will be merged into the system before the programs and d<\#230>mons are started.

Which files should be on the floppy? Consider the following list as a guide:

/etc/passwd/etc/shadow/etc/hosts/etc/hosts.allow/etc/hosts.deny/etc/resolv.conf/etc/HOSTNAME/etc/sysconfig/network-scripts/ifcfg-eth0/etc/sysconfig/network/etc/X11/XF86Config/etc/X11/X/etc/conf.modules

If you know nothing about these files or other configuration files, there is still hope for you. You must continue the boot and log in as root (I hope this will always be possible). The CD will be mounted in /mnt/cdrom. In /mnt/cdrom/system/config-touch you will find two scripts. Running **Touch_all** will touch all the files in the system. Then run the configuration tool you need (setup, xconfigurator or control-panel in X). Finally, run **Find_newer**. It will find the files updated by the configuration, and it will copy them to a directory named config in the current directory. Put the files on a floppy and you are done.

The configuration of Maragda Teaching is restricted. It does not ask for a floppy, and prompts you to choose one of four configurations already on the system:

1. PS2 mouse and frame buffer
2. serial mouse and frame buffer
3. PS2 mouse and vga
4. serial mouse and vga

In future versions of Maragda, I plan to detect the environment where Maragda is booting and auto configure it.

How to Build Maragda

There are three things to do:

1. Prepare a software installation containing all the packages you want on Maragda; we'll call it source installation.
2. Distribute the source installation between two big files, ROOT.FS and WHOLE.FS; they both should fit inside an ext2 file system.
3. Develop a boot mechanism in charge of detecting the CD-ROM drive and locating ROOT.FS on it. Then, it should load ROOT.FS as a RAM disk, preparing it to be the root file system, and finally it should leave the boot process to continue on the root file system.

All the necessary pieces that do the work (mainly shell scripts) are arranged into two working directories:

1. boot: in charge of developing the boot mechanism
2. system: contains the tools to create ROOT.FS and WHOLE.FS from the source installation

You can find those directories on the CD (along with the doc directory). They are a copy of my own working directories, and on them you will find all that is needed to build Maragda. You must copy them to a disk with at least 1GB of free space.

I should mention that, perhaps, some device files (files under /dev) might not be correct due to the copy from an ext2 file system to a CD file system (iso9660). In this case, you should replace those files by the corresponding /dev files from your system.

Now, let's outline the relevant steps in the building process and describe the purpose of each shell script as well as the main configuration files. Minor details are left out of the explanation; if you decide to build your own Maragda-like system, you should consult doc/developer.html on the CD and carefully read each shell script on the working directories.

Boot

Figure 1 shows the main steps to be taken in the boot process.

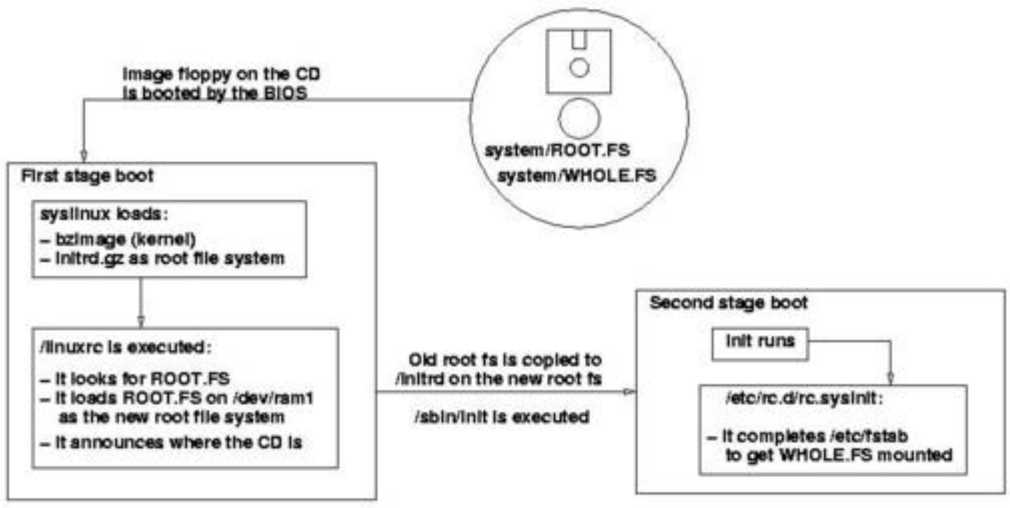


Image floppy on the CD is booted by the BIOS
system/ROOT.FS
system/WHOLE.FS
First stage boot
syslinux loads:
– bzimage (kernel)
– initrd.gz as root file system
/linuxrc is executed:
– It looks for ROOT.FS
– It loads ROOT.FS on /dev/ram1 as the new root file system
– It announces where the CD is
Old root fs is copied to /initrd on the new root fs
/sbin/init is executed
Second stage boot
init runs
/etc/rc.d/rc.sysinit:
– It completes /etc/fstab to get WHOLE.FS mounted

Figure 1. Main Steps of Boot Process

The target here is to create a bootable floppy disk that will be merged into the CD to make it bootable.

The boot loader for setting up the floppy is **syslinux**. It can boot a Linux system from a DOS-formatted floppy containing a kernel and an initial RAM disk file (compressed) with the root file system inside.

So, we'll need a kernel along with its loadable modules. (Only the kernel is required now.) I will not explain here how to compile a kernel but will only remark that it should support

- framebuffer device
- CD-ROM (iso9660 file system)
- initial RAM disk
- loop-back device

(You can find my kernel configuration on doc/developer.html.)

Now we have to build the initial RAM disk (initrd) holding the root file system for the first-stage boot. The easiest way to obtain one is to borrow it from a floppy rescue disk. But, I follow the “do it yourself” philosophy. The directory boot/initrd is a skeleton of the contents of my RAM disk. The script named **Init_initrd** fills that directory from scratch, copying files from the source installation. It creates directories, copies a subset of dev-files onto initrd/dev and populates initrd/bin. The size of an initrd must be kept small, so you should leave only the essential binaries on initrd/bin. Then you must run **Cp_lib**; it will copy the libraries on initrd/lib for the binaries you left before.

When you are finished, run **Make_initrd_fs_gz**. It will create a file (Minix formated), copying to it the files on boot/initrd and then compress it.

Now you have a kernel and an initial RAM disk. Next, you should generate the bootable floppy with syslinux. The syslinux configuration file, SYSLINUX.CFG, is contained in boot/syslinux_boot_floppy-20MB. The line:

```
append initrd=initrd.gz ramdisk_size=20480 vga=0x315
```

instructs the boot loader to load the initial RAM disk file on the floppy and passes two parameters to the kernel to inform it of the size of the RAM disk and the framebuffer video mode. The RAM disk size should be customized to the size of the ROOT.FS used.

Finally, the bootable floppy is created by the **Write_syslinux_boot_floppy** script. It will also extract a raw image of the floppy to be merged onto the CD.

The most relevant file here is boot/initrd/linuxrc (/linuxrc on the initial RAM disk) because this script is executed at boot up. It performs the main tasks intended for this stage:

- It searches for the device holding the files:

```
/MARAGDA
/system/ROOT.FS
/system/WHOLE.FS
```

- If that is found, it then states which device will be the new root file system:

```
mount -n /proc /proc -t proc
echo 0x101 > /proc/sys/kernel/real-root-dev
(0x101 represents /dev/ram1)
```

- It dumps WHOLE.FS on /dev/ram1.
- Finally, it writes on **cdrom_dev** which device the .FS files were found. This file will be consulted during the second-stage boot.

If everything goes okay, the boot process will continue by executing **init**, already in the new root file system. The old file system (initrd) will remain mounted under /initrd on the new one.

ROOT.FS and WHOLE.FS

These files are created by the scripts on the system working directory and the files found there, starting with the source installation.

Cp_root is in charge of creating ROOT.FS, while **Cp_whole** will build WHOLE.FS. Both scripts use **Make_fs** to create file systems inside a file.

ROOT.FS should contain files and directories that need to be written and cannot live on a read-only file system. Writing will be possible since ROOT.FS will reside on a RAM disk. It is difficult to find out exactly what is required to be in ROOT.FS. After several attempts, I decided to copy at least the following directories: bin, sbin, lib (including lib/modules), home, var, etc.

For those directories notcopied, symbolic links are created, pointing to where WHOLE.FS will be mounted once Maragda is running (/mnt/whole_fs).

In addition, binaries and libraries are unstripped (symbols removed) to reduce space.

The configuration files for ROOT.FS are kept under system/config. Once ROOT.FS is populated from the source installation, system/config files are written to ROOT.FS overwriting existing files. Typical system/config files are:

- the /home/jordi directory
- /etc/X11/X and /etc/X11/XF86Config for configuring X
- /etc/hosts* and /etc/sysconfig/network-scripts, for setting up network
- /etc/passwd, for users' accounts
- /etc/rc.d/init.d/* files, to drive the startup process

Especially important is the script system/config/etc/rc.d/rc.sysinit. It is run by init (in the second-stage boot). I patched it to complete the Maragda system. It adds some lines to Maragda's /etc/fstab to reflect where the CD is and, therefore, where WHOLE.FS is. Remember that this was announced by linuxrc (first-stage boot) on the file /initrd/cdrom_dev. Once fstab is completed, rc.sysinit goes on with the “side effect” that WHOLE.FS will be correctly mounted.

Doing Tests

Of course, to test your emerging system, you do not need to record a CD. It can be booted easily by means of LILO, arranging files this way:

```
/system/MARAGDA
/system/ROOT.FS
/system/WHOLE.FS
/boot/bzImage-rd (the kernel)
/boot/1INITRD.GZ (the initial RAM disk for the first stage boot)
```

(System may be a symbolic link to the working directory.) You should add the following entry to **/etc/lilo.conf**:

```
image=/boot/bzImage-rd
label=maragda
initrd=/boot/1INITRD.GZ
literal="ramdisk_size=20480" # size of ROOT.FS
read-only
vga=0x315
```

and run LILO every time you make any change.

The Source Installation

As you have already learned, the source installation is a directory containing a complete, installed Linux system. It can be the source for copying the ROOT.FS and WHOLE.FS files or even for installing Linux onto a hard-disk partition or onto a 250MB zip floppy. The script **Prepare_install** (under the system working directory) sets up a directory to receive RPM packages. Then, you can use **Install_rpm** to install those packages onto the source-installation directory.

Install_rpm installs packages from the Red Hat's 6.2 distribution CD (or from another place, if you change it). The names of packages to be installed are grouped in files under system/RPM_lists. For instance, base_inst lists the packages for the base-system installation. I created the listing files starting with the file RedHat/base/comps on the Red Hat CD.

If you install the packages in the right order (as it appears on Install_rpm) they should be installed with no problem. Install_rpm also reports which packages were not installed. In the case of base_inst it creates base_inst_not_installed. This is the only group of rpms that poses a problem. But, if you tell Install_rpm to install packages on base_inst_not_installed, finally all packages will be installed.

You may find all the scripts used on http://navel.eugan.upv.es/maragda/doc, including raw CD images of Maragda (and the instructions to record them).

Enjoy it.

Jordi Bataller was born in Ador (a little town whose people cultivate orange trees) thirty years ago. He has been studying Linux since 1992 and hopes to do so for life. He works as a professor at the Polytechnical College of Gandia and as researcher at the Information Technology Institute, both at the Polytechnical University of Valencia (Spain) where he received his PhD. In his spare time he enjoys swimming and playing the clarinet.

Archive Index Issue Table of Contents

Advanced search

Designing and Using DMZ Networks to Protect Internet Servers

Mick Bauer

Issue #83, March 2001

Mick explains how to care for services that come into contact with untrusted networks.

One of the most useful tools in firewall engineering today is the DMZ, or DeMilitarized Zone, a network where all publicly accessible services are placed so they can be more closely watched and, also, isolated from one's internal network. DMZs, bastion servers and Linux make a particularly good combination.

But what, really, is a DMZ? Is there more than one correct way to design one? Does everyone who hosts internet services need a DMZ network? These are issues I really haven't addressed yet, so this month we're going to take a higher-level look at DMZ security.

By the way, you may decide that your current DMZ-less firewall system is reasonable for your needs. I hope you keep reading, regardless: any host or service (whether on a DMZ or not) that has direct contact with untrusted networks demands particular care, and many of the techniques and considerations discussed in this article apply to both non-DMZ and DMZ environments.

Some Terminology

Let's get some definitions cleared up before we proceed. These may not be the same definitions you're used to or prefer, but they're the ones I use in this article:

- *DMZ (DeMilitarized Zone)*: a network containing publicly accessible servers that is isolated from the “internal” network proper but not necessarily from the outside world.
- *Internal Network*: that which we're trying to protect: end-user systems, servers containing private data and all other systems with which we do not wish the outside world to initiate connection. Also called the protected network.
- *Firewall*: a system or network that isolates one network from another. This can be a router, a computer running special software in addition to or instead of its standard operating system, a dedicated hardware device (although these tend to be prepackaged routers or computers), or any other device or network of devices that performs some combination of packet filtering, application-layer proxying and other access control. In this article the term will generally refer to a single multihomed host.
- *Multihomed Host*: any computer having more than one network interface.
- *Bastion Host*: a system that runs publicly accessible services but is not itself a firewall. Bastion Hosts are what we put on DMZs (although they can be put anywhere). The term implies that a certain amount of OS-hardening has been done, but this (sadly) is not always the case.
- *Packet Filtering*: inspecting the IP headers of packets and passing or dropping them based on some combination of their Source IP Address, Destination IP Address, Source Port (Service) and Destination Port (Service). Application data is not considered, i.e., intentionally malformed packets are not necessarily noticed, assuming their IP headers can be read. Packet filtering is part of nearly all firewalls' functionality but is not considered, in and of itself, to be sufficient protection against any but the most straightforward attacks. Most routers (and many low-end firewalls) are limited to packet filtering when it comes to network security.
- *Proxying*: to act as an intermediary in all interactions of a given service type (FTP, HTTP, etc.) between internal hosts and untrusted/external hosts. This implies, but does not guarantee, sophisticated inspection of Application-Layer data (i.e., more than simple packet filtering). Some firewalls possess, and are even built around, Application-Layer Proxies. Each service to be proxied must be explicitly supported (i.e., “coded in”); firewalls that rely on Application-Layer Proxies tend to use packet filtering or rewriting for services they don't support by default.

- *Stateful Inspection*: at its simplest, this refers to the tracking of the three-way handshake (host1:SYN, host2:SYNACK, host1:ACK) that occurs when each session for a given TCP service is initiated. At its most sophisticated, it refers to the tracking of this and subsequent (including application-layer) state information for each session being inspected. The latter is far less common than the former.

That's a mouthful of jargon, but it's useful jargon (useful enough, in fact, to make sense of the majority of firewall-vendors' propaganda). Now we're ready to dig into DMZ architecture.

Types of Firewall and DMZ Architectures

In the world of expensive commercial firewalls (the world in which I earn my living), the term firewall nearly always denotes a single computer or dedicated hardware device with multiple network interfaces. Actually, this definition can apply to much lower-end solutions as well: network interface cards are cheap, as are PCs in general.

Regardless, this is different from the old days when a single computer typically couldn't keep up with the processor overhead required to inspect all ingoing and outgoing packets for a large network. In other words, routers, not computers, used to be the first line of defense against network attacks.

This is no longer the case. Even organizations with high-capacity Internet connections typically use a multihomed firewall (whether commercial or OSS-based) as the primary tool for securing their networks. This is possible thanks to Moore's law, which has provided us with inexpensive CPU power at a faster pace than the market has provided us with inexpensive Internet bandwidth. In other words, it's now feasible for even a relatively slow PC to perform sophisticated checks on a full T1's-worth (1.544MBps) of network traffic.

The most common firewall architecture one tends to see nowadays, therefore, is the one illustrated in Figure 1. In this diagram, we have a packet-filtering router that acts as the initial but not sole line of defense. Directly behind this router is a proper firewall, in this case a Sun SparcStation running, say, Red Hat Linux with IPChains. There is no direct connection from the Internet or the external router to the internal network: all traffic to it or from it must pass through the firewall.

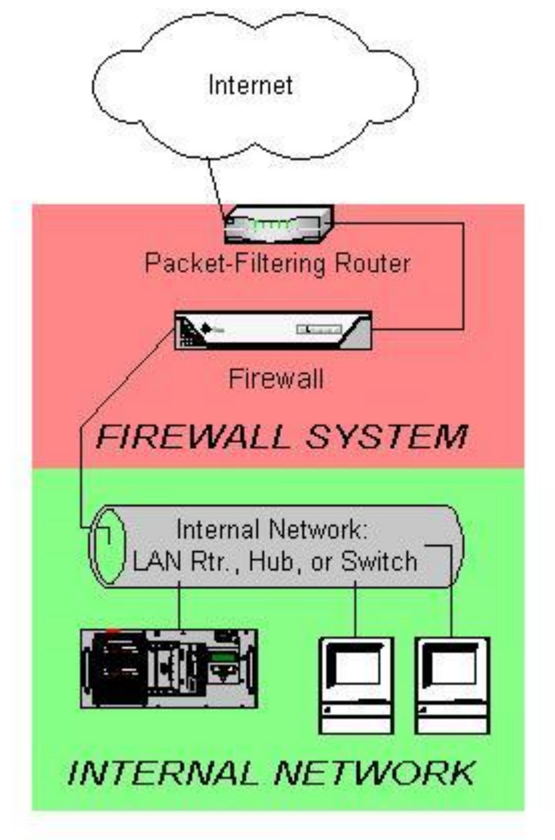


Figure 1: "Multi-Homed Host" Firewall

Figure 1. “Multihost Firewall”

By the way, in my opinion, all external routers should use some level of packet filtering (aka “Access Control Lists” in the Cisco lexicon). Even when the next hop inward from such a router is an expensive and/or carefully configured and maintained firewall, it never hurts to have redundant enforcement points. In fact, when several Check Point vulnerabilities were demonstrated at the most recent Black Hat Briefings, no less a personage than a Check Point spokesperson mentioned that it's foolish to rely solely on one's firewall!

What's missing or wrong in Figure 1? (I said this architecture is common, not perfect!) Public services such as SMTP (e-mail), Domain Name Service (DNS) and HTTP (WWW) must either be sent through firewall to internal servers or hosted on the firewall itself.

Passing such traffic doesn't automatically expose other internal hosts to attack, but it does magnify the consequences of such a server being compromised. Hosting public services on the firewall isn't necessarily a bad idea on the face of it, either (what could be a more secure environment than a firewall?), but the performance issue is obvious: the firewall should be allowed to use all its available resources for inspecting and moving packets. (Although there are some possible exceptions that we'll examine shortly.)

Where, then, to put public services so that they don't directly or indirectly expose the internal network and overtax the firewall? In a DMZ network, of course! At its simplest, a DMZ is any network reachable by the public but isolated from one's internal network. Ideally, however, a DMZ is also protected by the firewall. Figure 2 shows my preferred firewall/DMZ architecture.

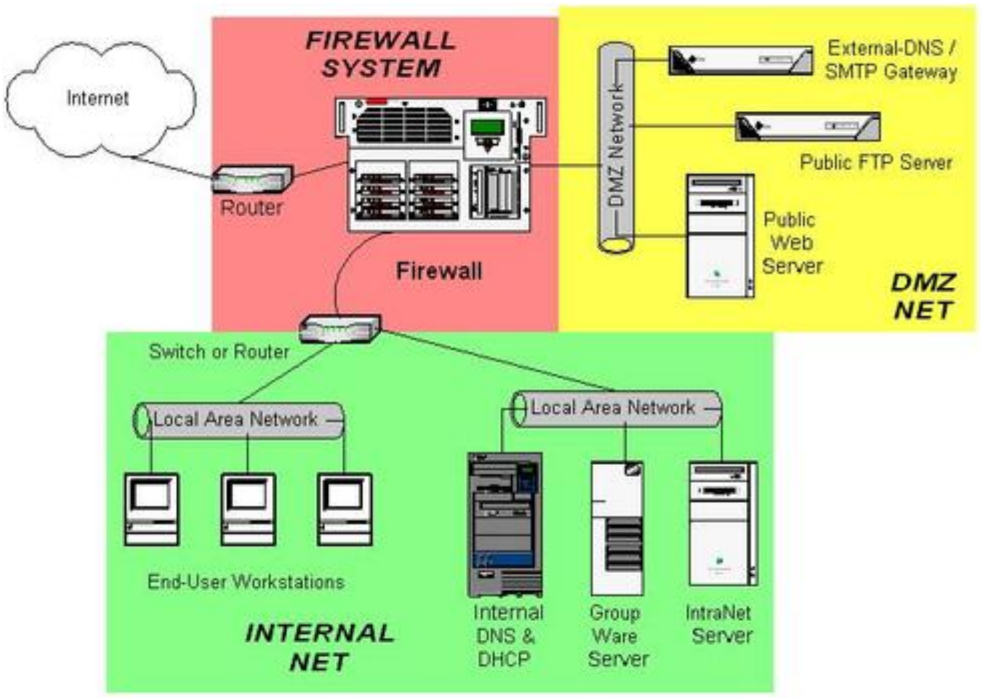


Figure 2: "Multi-Homed Host" Firewall With DMZ

Figure 2. “Multihoned Host” Firewall with DMZ

In Figure 2 we have a *three*-homed host as our firewall, placed so that hosts providing publicly accessible services are in *their own* network with a dedicated connection to the firewall, with the rest of the corporate network facing a different firewall interface. If configured properly, the firewall uses different rules in evaluating traffic from the Internet to the DMZ, from the DMZ to the Internet, from the Internet to the internal network, from the internal network to the Internet, from the DMZ to the internal network and from the internal network to the DMZ.

This may sound like more administrative overhead than with internally-hosted or firewall-hosted services, but actually, it's potentially much simpler because the DMZ can be treated as a single entity. In the case of internally hosted services, each host must be considered individually unless they're all located on a single IP network otherwise isolated from the rest of the internal network.

Other architectures are sometimes used, and Figure 3 illustrates two of them. The Screened Subnet architecture is completely dependent on the security of both the external and internal routers. There is a direct physical path from the outside to the inside, a path controlled by nothing more sophisticated than the router's packet-filtering rules.

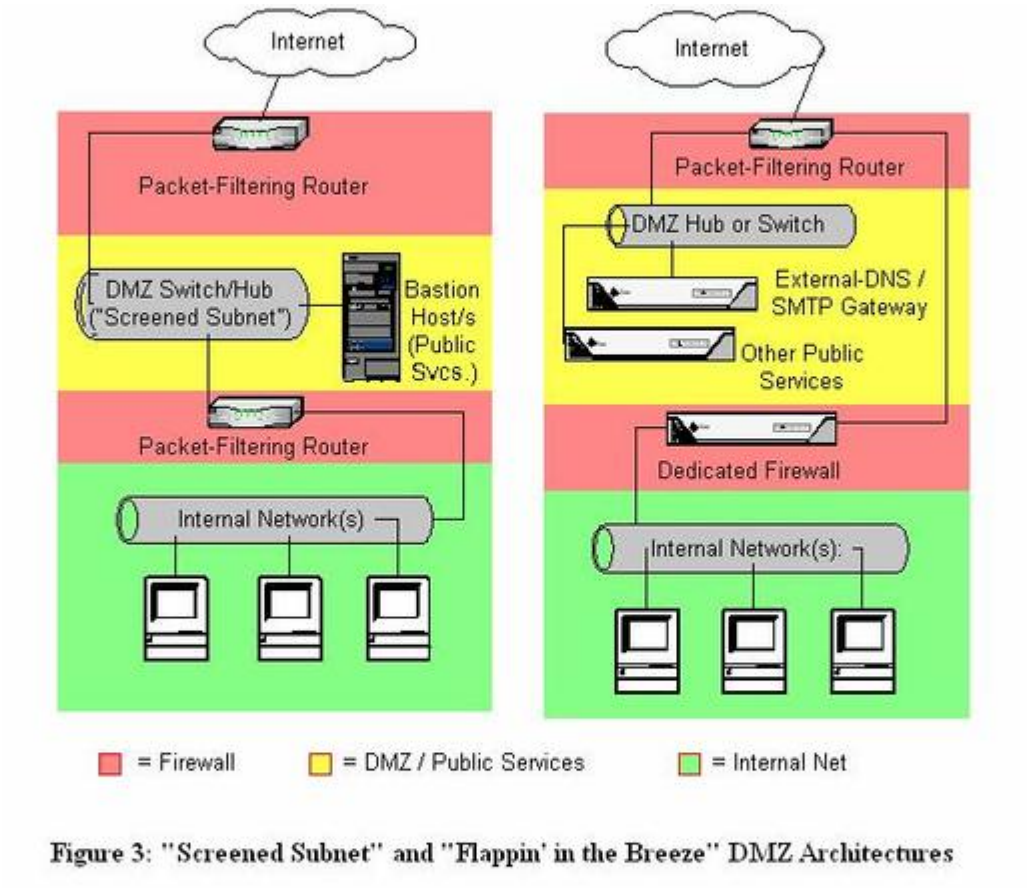


Figure 3. “Screend Subnet” and “Flapping in the Breeze” DMZ Architectures

The right-hand illustration in Figure 3 shows what I call the “Flapping in the Breeze” DMZ architecture, in which there *is* a full-featured firewall between the Internet and the internal network but *not* between the Internet and the DMZ, which is placed *outside* of the firewall and is protected only by a single packet-filtering router.

Both the Screened Subnet and Flapping in the Breeze architectures still show up in firewall textbooks (albeit with different names), but in my opinion, they both place too much trust in routers. Such trust is problematic for several reasons: first, in may organizations routers are under a different person's control than the firewall is, and this person many insist that the router have a weak administrative password, weak access-control lists or even a modem attached so that the router's vendor can maintain it; second, routers are considerably more hackable than well-configured computers (for example, by default they nearly always support remote-administration via Telnet, a highly insecure service); and third, packet filtering is a crude and incomplete means of regulating network traffic.

Even an OSS/freeware-based firewall can support IPSEC, application-layer proxies, stateful inspection, RADIUS authentication and a variety of other sophisticated controls unavailable on most routers. When all is said and done, routers are designed to route, not to protect.

What about Cisco PIX? The PIX firewall *is* a router but with a hardened and security-focused version of the Cisco IOS operating system. Although it relies heavily on simple packet filtering, it supports enough additional features to be a good firewall if properly configured. When I question the viability of routers as firewalls, I'm referring to nonhardened, general-purpose routers.

In summary, what one's DMZ architecture looks like depends on what one's firewall architecture looks like. A firewall design built around a multihomed host lends itself to the DMZ architecture I recommend (see Figure 2), in which the DMZ is connected to its own interface on the firewall host and, thus, is isolated from both the Internet and one's internal network.

Deciding What Should Reside on the DMZ

Once you've decided where to put the DMZ, you need to decide precisely what's going to reside there. In a nutshell, my advice is to put *all* publicly accessible services in the DMZ.

Too often I encounter organizations in which one or more crucial services are passed through the firewall to an internal host despite an otherwise strict DMZ policy. Frequently, the exception is made for MS-Exchange or some other application not necessarily designed for Internet-strength security to begin with and which hasn't been hardened to the extent it could be.

But, the one application passed through in this way becomes the dreaded single point of failure: all it takes is one buffer-overflow vulnerability in that application for it to be possible for an unwanted visitor to gain access to all hosts reachable by that host. Far better for that list of hosts to be a short one, i.e., DMZ hosts, than a long one (and a sensitive one), i.e., all hosts on the internal network. This point can't be stressed enough: the real value of a DMZ is that it allows us to better manage and contain the risk that comes with Internet connectivity.

Furthermore, the person who manages the passed-through service may be different from the one who manages the firewall and DMZ servers and may not be quite as security-minded. If for no other reason, all public services should go on a DMZ so that they fall under the jurisdiction of an organization's most paranoid system administrators, namely, its firewall administrators.

Does this mean that the corporate e-mail, DNS and other crucial servers should all be moved from the inside to the DMZ? Absolutely not! They should instead be split into internal and external services (see Figure 2).

DNS, for example, should be split into external DNS and internal DNS. The external DNS zone information, which is propagated out to the Internet, should contain only information about publicly accessible hosts. Information about other, nonpublic hosts should be kept on separate internal DNS zone lists that can't be transferred to or seen by external hosts.

Similarly, internal e-mail (i.e., mail from internal hosts to other internal hosts) should be handled strictly by internal hosts, and all Internet-bound or Internet-originated mail should be handled by a DMZ host, usually called an SMTP Gateway. (For more specific information on Split-DNS servers and SMTP Gateways and how to use Linux to create secure ones, see the October 2000 issue of *Linux Journal*.)

Thus, almost any service that has both private and public roles can and should be split in this fashion. While it may seem like a lot of added work, it need not be, and in fact, it's liberating: it allows you to construct your internal services based primarily on features or other management- and end-user-friendly factors while designing your public (DMZ) services based primarily on security and performance factors. It's also a convenient opportunity to integrate Linux, OpenBSD and other open-source software into otherwise proprietary-software-intensive environments!

Needless to say, any service that is strictly public (i.e., not used in a different or more sensitive way by internal users than by the general public) should reside solely in the DMZ. In summary: all public services, including the public components of services that are also used on the inside, should be split if applicable and hosted in the DMZ, without exception.

Allocating Resources in the DMZ

Okay, so everything public goes in the DMZ. But does each service need its own host? Can any of the services be hosted on the firewall itself? And should one use a hub or a switch on the DMZ?

The last question is the easiest: with the price of switched ports decreasing every year, switches are preferable on any LAN and especially so in DMZs. Switches are superior in two ways. From a security standpoint, they're better because it's impossible to sniff or eavesdrop traffic not delivered to one's own switch-port. Because one of our assumptions about DMZ hosts is they are more likely to be attacked than internal hosts, this is important. We need to think not only about how to prevent these hosts from being compromised but also about what the consequences might be if they are—being used to sniff other traffic on the DMZ is one possible consequence.

The other way switches are better than hubs is, of course, their performance. Most of the time each port has its own chunk of bandwidth rather than sharing one big chunk with all other ports. Note, however, that each switch has a backplane that describes the actual volume of packets the switch can handle; a 10-port 100Mbps switch can't really process 1000MBps if it has an 800MBps backplane. Even so, low-end switches disproportionately outperform comparable hubs.

Still, a hub may yet suffice for your DMZ depending on the number of hosts on your DMZ, the degree to which you're worried about a compromised DMZ host being used to attack other DMZ hosts and the amount of traffic to and from the DMZ.

The other two questions can usually be determined by nonsecurity-driven factors (cost, expected load, efficiency, etc.) *provided* that all DMZ hosts are thoroughly secured and monitored. Also, firewall rules (packet filters, etc.) governing traffic to and from the DMZ need to be as paranoid as possible.

Guidelines for Securing DMZ Hosts

It would seem to be common sense that each host on a DMZ must be thoroughly nailed down. But sure enough, one commonly encounters organizations paranoid (prudent) enough to have a DMZ but not quite paranoid enough to secure their DMZ properly. The good news is that with a little time and a properly suspicious attitude, you can significantly lower any system's exposure to compromise by script kiddies.

Always run the latest stable version of your operating system, software and kernel, and keep current with security patches as they are released.

If everyone followed this simple and obvious tenet, the “Rumors from the Underground” list of hacked web pages on www.hackernews.com would be a lot shorter. As we discussed last month in “Securing DNS”, the vast majority of DNS-based hacks don't apply to the most recent versions of BIND; the same can safely be said for most other Linux network software packages. We all know this, but we don't always find the time to follow through.

Disable all unnecessary services and dæmons.

A program you don't use for anything important is a program you've got little reason to maintain properly and is, therefore, an obvious target for attackers.

This is an even easier thing to fix than old software. At setup time, simply delete or rename all unneeded links in the appropriate runlevel directory in /etc/rc.d/.

For example, if you're configuring a web server that doesn't need to be its own DNS server, you would enter something like the following:

```
mv /etc/rc.d/rc2.d/S30named /etc/rc.d/rc2.d/disabled_S30named
```

(Note that your named startup script may have a different name and may exist in different or additional subdirectories of /etc/rc.d.)

While any unneeded service should be disabled, the following deserve particular attention:

- **RPCservices**: Sun's Remote Procedure Control protocol (which is included nowadays on virtually all flavors of UNIX) lets you execute commands on a remote system via **rsh**, **rcp**, **rlogin**, **nfs**, etc. Unfortunately, it isn't a very secure protocol, especially for use on DMZ hosts. You shouldn't be offering these services to the outside world—if you need their functionality, use **ssh** (the Secure Shell), which was specifically designed as a replacement for rpc services. Disable (rename) the nfsd and nfsclientd scripts in all subdirectories of /etc/rc.d in which they appear, and comment out the lines corresponding to any r-commands in /etc/inetd.conf. (Warning: local processes sometimes require the RPC portmapper, aka rpcbind—disable this with care, and try re-enabling it if other things stop working).
- **inetd**: The Internet Dæmon is a handy way to use a single process (i.e., inetd) to listen on multiple ports and to invoke the services on whose behalf it's listening on an as-needed basis. Its useful life, however, is drawing to a close: even with TCP Wrappers (which allow one to turn on very granular logging for each inetd service), it isn't as secure as simply running each service as a dæmon. (An FTP server really has no reason not to be running FTPD processes all the time.) Furthermore, most of the services enabled by default in inetd.conf are unnecessary, insecure or both. If you must use inetd, edit /etc/inetd.conf to disable all services you don't need (or never heard of). Note: many rpc services are started in inetd.conf.
- **linuxconfd**: While there aren't any known exploitable bugs in the current version of linuxconf (a system administration tool that can be accessed remotely), CERT reports that this service is commonly scanned for and may be used by attackers to identify systems with other vulnerabilities (CERT Current Activity page 7/31/2000, www.cert.org/current/current_activity.html).

- **sendmail**: Many people think that sendmail, which is enabled by default on most versions of UNIX, is necessary even on hosts that send e-mail only to themselves (e.g., administrative messages such as Ctrl+tab output sent to root by the crontab dæmon). This is not so: sendmail (or postfix, qmail, etc.) is needed only on hosts that must deliver mail to or receive mail from other hosts. sendmail is usually started in /etc/rc.d/rc2.d or /etc/rc.d/rc3.d.
- **Telnet**, **FTP** and **POP**: These three protocols have one very nasty characteristic in common: they require users to enter a username and password that are sent in clear text over the network. Telnet and FTP are easily replaced with ssh and its file-transfer utility **scp**; e-mail can either be automatically forwarded to a different host, left on the DMZ host and read through a ssh session or downloaded via POP using a “local forward” to ssh (i.e., piped through an encrypted Secure Shell session). All three of these services are usually invoked by inetd. These dæmons are usually started by inetd.

Run services “chrooted” whenever possible.

Some dæmons, such as named, have explicit support for being run in a “chroot jail” (i.e., such that to the chrooted process, “/” is actually some other directory that can't be navigated out of). This is a valuable security feature; if a chrooted process is hijacked or exploited somehow, the attacker will not be able to access files outside of the chroot jail.

On Linux, even processes that don't have built-in chroot support can be run chrooted: simply type **chroot chroot-jail path command string.** For example, to run the imaginary command bubba -v plop chrooted to /var/bubba, you would type:

```
chroot /var/bubba /usr/local/bin/bubba -v plop
```

Note, however, that any system files a chrooted process needs in order to run must be copied to appropriate subdirectories of the chroot jail. If our imaginary process bubba needs to parse /etc/passwd, we need to put a copy of the passwd file in /var/bubba/etc. The copy need not, however, contain any more information than the chrooted process needs; to extend our example still further, if bubba is a server that only anonymous users may access, /var/bubba/etc/passwd probably only needs one line (e.g., **nobody::50:50:Anonymous user::/bin/noshell**).

Run services with unprivileged UIDs and GIDs whenever possible.

While some dæmons will only work if run by root (the default UID of processes invoked at startup time), nowadays many programs can be set to run as unprivileged users. For example, Postfix, Wietse Venema's sendmail replacement, usually runs with a special, unprivileged account named postfix.

This has a similar effect as chroot (and in fact the two often go together). Should the process become hijacked or otherwise compromised, the attacker will gain a level of access privileges lower than root's (hopefully much lower). Be careful, however, to make sure that such an unprivileged account still has enough privilege to do its job.

Delete or disable unnecessary user accounts.

Some Linux distributions have, by default, lengthy /etc/passwd files that contain accounts even for use by software packages that haven't been installed. My laptop computer, for example, which runs SuSE Linux, has 22 unnecessary entries in /etc/passwd. Commenting out or deleting such entries, especially the ones that include executable shells, is important.

Configure logging and check logs regularly.

This is another thing we all know we should do but often fail to follow through on. You can't check logs that don't exist, and you can't learn anything from logs you don't read. Make sure your important services are logging at an appropriate level, know where those logs are stored and whether/how they're rotated when they get large, and get in the habit of checking the current logs for anomalies.

grep is your friend here: using **cat** alone tends to overwhelm people. You can automate some of this log-parsing with shell scripts; scripts are also handy for running **diff** against your system's configuration files to monitor changes (i.e., by comparing current versions to cached copies).

If you have a number of DMZ hosts, you may wish to consider using syslogd's ability to consolidate logs from several hosts on one system. You may not realize it, but the syslog dæmon can be configured to listen not only for log data from other processes on the local system, but on data from remote hosts as well. For example, if you have two DMZ hosts (bobo and rollo) but wish to be able to view both machines' logs in a single place, you could change bobo's /etc/syslogd.conf to contain only this line:

```
  *.*        @rollo
```

This will cause syslogd on bobo to send all log entries not to its own /var/log/messages file but to rollo's.

While handy, be aware that this technique has its own security ramifications: if rollo is compromised, bobo's logs can also be tampered with. Furthermore, rollo's attacker may learn valuable information about bobo that they can subsequently use to attack bobo. This may or may not be of concern to you, but you should definitely think about whether the benefit justifies the exposure (especially given that the benefit may be that you can more effectively prevent your DMZ hosts from being compromised in the first place).

We'll close with the guideline that makes DMZs worthwhile in the first place.

Use your firewall's security policy and anti-IP-spoofing features.

Naturally, you want to carefully restrict traffic from the outside world to the DMZ. But it's equally important to carefully restrict traffic from the DMZ to your internal network (to protect it in the event that a DMZ host is compromised) and from the DMZ to the outside world (to prevent a compromised DMZ host from being used to attack other networks).

It goes without saying that you'll probably want to block all traffic from the Internet to internal hosts. (You may or may not feel a need to restrict traffic from the internal network to the DMZ, depending on what type of access internal users really need to DMZ hosts and how much you trust internal users.) In any event, your firewall-security policy will be much more effective if your firewall can distinguish between legitimate and phony source-IP addresses. Otherwise, it might be possible for an external user to slip packets through the firewall by forging internal source IPs.

By default, most firewalls don't have this functionality enabled (the feature is usually called something like anti-IP-spoofing. Even if your firewall supports it, you'll probably have to configure and start it yourself. It's well worth the effort, though.

Mick Bauer (mick@visi.com) is security practice lead at the Minneapolis bureau of ENRGI, a network engineering and consulting firm. 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

Focus on Software

David Bandel

Issue #83, March 2001

MonMotha IPTables Firewall script, poppy, Apache Toolbox and more.

Consulting in the Linux arena today is a challenge. I've worked on a number of different UNIX systems: Solaris, SunOS, Ultrix, OpenServer, AIX, HP-UX and, of course, Linux. What's the difference between all the flavors of UNIX I just mentioned and Linux? Well, the biggest difference is probably that given any flavor of UNIX, all installations are basically the same. The installation, system administration, etc., is uniform. With Linux each distribution has its installation routines, system administration scripts and ways of handling things. Fortunately, they're all Linux, so apart from a version number, they're pretty much all the same. DNS is the same; DHCP is the same; Apache is the same, and, of course, the kernel itself is the same. If you are or want to be a Linux consultant, I highly suggest you learn Linux from the bottom up (i.e., the CLI, command-line interface) rather than from the top down (X server and the admin tools peculiar to a given distro). Learn to read shell scripts and follow their workings. Find, install and use distribution-neutral tools where possible, like webmin (which can be secured via SSL). After all, if you can read **/etc/named.conf**, DNS is easy to tackle. Ditto for just about every other service on the system. I don't install webmin just for me, I install it for my clients. In fact, I don't actually use it. I think you'll have fewer headaches with this approach, and then your clients can use whichever distribution strikes their fancy (or is most appropriate) rather than reinstalling all the Linux systems because you're unfamiliar with the one being used. Anyway, it works for me.

MonMotha IPTables Firewall script: t245.dyndns.org/~monmotha/firewall/index.php

I don't usually recommend firewall tools or firewall scripts. In fact, I'm not really recommending this one per se. But by the time you read this, several distros should be out with the new 2.4 kernel and netfilter. This particular firewall script can help you if you're having problems getting started with iptables. It makes a good baseline and is a good compromise. The author takes advantage of the stateful capability of netfilter, but I'd add to it. Again, while a good start, you really should look it over and incorporate any necessary changes for your particular situation. Requires: iptables, sh.

poppy: home.sprynet.com/~cbagwell/projects.html

A universal (or pretty close) command-line mail program that will read your mail from a POP3 or IMAP server. This is really good. I often find myself on the end of a slow dial-up with some large mail messages. With this program I can quickly look at subjects or go through each e-mail one by one deleting, replying or leaving until later. Requires: Perl.

Apache Toolbox: http://www.apachetoolbox.com/

Do you need to get a working Apache with several mods up and running quickly and correctly? Never compiled Apache before? If you answered yes to both questions, you have a recipe for failure. But relax! The Apache Toolbox will help you. It even knows (and will warn you about) the mod_perl/php4 clash that causes segfaults. I've compiled and installed custom Apache installations a number of times, but it doesn't get any easier than this. While not infallible, it does a better job than all but the most experienced Apache builder. It won't, however, allow you to install both php3 and php4—you'll have to add one of them later if you want both. Requires: sh, wget.

indexpage: http://www.lysator.liu.se/~unicorn/hacks/indexpage/

Do you have a lot of pictures (jpeg) you'd like to put up on a web site quickly? I had a whole directory full of grabs from my CamCorder. I just slipped this Perl program into the directory, quickly worked up a descriptions file, ran the program and *voilà*: four html pages of thumbnails. The image size doesn't matter: the program will stretch, shrink or otherwise make the image fit into the box. If you make any changes (add, subtract or just move pictures around), just running the program again will recreate the pages. Requires: ImageMagick, Perl, Perl modules: Image::Size.

tcpspy: http://box3n.gumbynet.org/~fyre/software/

Need to find out who's connecting where, when and how on your system? This program will provide you with a log of connections, disconnections, users, local IP:port pairs, remote IP:port pairs and even programs. Or maybe you don't want to know. It sure is interesting finding out who's running nmap against which targets, when from your system. Because it uses syslog, that information can be sent to your central-logging server. By default, tcpspy uses the log facility LOCAL1, but you can change that in the Makefile to just about facility. Requires: glibc.

BlackNova Traders: http://blacknova.net/

Here's a space strategy game for web play. The object: trade and claim planets ultimately to win by owning more “stuff”. And if you are killed, you lose. It's a great game for those who enjoy text-based games. No fancy graphics, just trading, occupying and protecting planets and trying not to be killed by anyone else. Requires: Apache with PHP3 with MySQL support, MySQL, web browser, cron.

plbackitup: www.glandrake.com/scripts.html

This Perl script will allow you to back up whatever directories you want and permit you to exclude files and directories beneath the directory to be backed up. This backup creates a local file. If you want to store the backup elsewhere, you can FTP it to a central storage server. With several systems using a central storage server (the only system I have a tape drive installed on), I find this utility very handy. Requires: Perl, standard UNIX tools (tar, touch, rm, others).

Bug Tracker: www.agstools.com/products/bt.html

If you need to track various projects as well as bugs, workarounds and enhancements on those projects, this might be what you need. This application is easily installed and can be accessed from anywhere via a web browser. Users (developers) log in with their e-mail addresses and a password to access the bug database. Works well from a variety of browsers (as long as they have support for cookies for logins). Requires: Database (PostgreSQL, MySQL), Perl, Perl modules CGI, DBI, DBD::, web browser.

Until next month.

David A. Bandel (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is coauthor of Que Special Edition: Using Caldera OpenLinux.

Archive Index Issue Table of Contents

Advanced search

An Interview with Greg Haerr on the Past, Present and Future of Microwindows

Rick Lehrbaum

Issue #83, March 2001

Haerr gives specifics on the Microwindows Project.

Greg Haerr is CEO of Century Software and founder of the Microwindows Project, a GUI/windowing solution for embedded systems and devices. In this interview, Haerr relates how he came to be working with embedded Linux, explains why he created Microwindows, offers a brief comparison between Microwindows and alternative embedded GUIs, contemplates the future of Microwindows and offers his perspective on open source.

Rick: When were you first exposed to Linux, and what led you to begin developing GUI/windowing software for Linux?

Greg: My involvement with Linux goes back to 1993, to one of the first 386 distributions, Yggadrisil. I remember that this distribution came with a boot floppy and a CD, which was new for the time. On bootup, the system ran the UMSDOS file system, which allowed Linux to run on top of the MS-DOS system's free space, and then proceeded to run the X Windows System, display a login prompt and play the theme to *Star Trek*. The display would switch between three graphics resolutions when you hit the Alt-Plus key...Linux has always shown from the start its extreme adaptability to differing hardware environments and its ability to take maximal advantage of that hardware with minimal user reconfiguration.

I've been programming for over 25 years, long before I became a businessman. Although I've followed Linux for seven years, it's only been during the last two years that I've contributed heavily to important projects. It's definitely been worth it.

Rick: To date, the greatest popularity of Linux has been in the area of web servers and Internet infrastructure systems. Why do you feel Linux is a good choice for embedded systems?

Greg: Why run Linux on embedded devices? In my estimation, there are at least four compelling reasons that show that Linux will continue to grow extremely rapidly in the next decade. First, the 32-bit microprocessor has finally come down in price and power requirements, and up in speed so that it can be used very cost effectively in the embedded devices desired today, including handheld computers, webpad-like devices and devices running flat-panel displays. And Linux nearly from the start has been a multiple-architecture operating system, which already supported these advanced RISC processors. So the convergence of Linux and embedded is quite natural.

Secondly, and especially in the graphical area in which I'm focused, software developers and systems architects want to leverage the tremendous success that has occurred on the desktop to more portable devices. There's no need to reinvent the wheel when so many applications that have propelled the desktop forward can now be used to enable wireless handheld computing.

Finally, Linux, being royalty-free and open source, allows the manufacturers to control their costs during the development phase, while sharing inventions and algorithms allows the best implementations to be deployed, which is a win-win for producers and consumers alike.

Rick: How does Linux compare to the “traditional” operating systems that have been used in the embedded market? That is, OSes like VxWorks, pSOS, VRTX, etc.

Greg: One of the key advantages Linux has over other choices comes in the development cycle: developers run exactly the same operating system on their desktop as they deploy on the target embedded device. This means that they are as completely familiar with the features and capabilities of the target device as they are of the desktop. For instance, in Century Software's new Microwindows operating environment and development toolkit for the Compaq iPAQ, we can completely simulate the target device graphical screen layout on the desktop host environment, thus allowing the applications to be built and tested while the hardware is also being developed.

Of course there are other advantages too, especially for those customers that already have a considerable investment in UNIX, whether it be from Sun, DEC, HP or IBM. And that is if you've got applications that you've developed for the UNIX environment and want to take advantage of the newly available capabilities of handheld and wireless technology, it's unbelievably more simple to deploy those UNIX applications on Linux than it is to try to run them on say, Windows CE. And of course since the majority of web servers are already running Linux, it's natural to build client-side applications using the Linux operating system.

Rick: What made you begin developing Microwindows? What did you have in mind when you started the project—what were your goals?

Greg: Well, I designed the Microwindows windowing environment initially for fun, since I have a love of small systems doing big things. However, this quickly turned into a project where user interest drove me to implement features found only on more advanced systems, but in a small space, without having to be ultimately general. For instance, Microwindows supports anti-aliased text, TrueType and Adobe Type 1 fonts and alpha blending, all of which X is just getting around to support.

Microwindows, when compared to X and including all the libraries, is still an order of magnitude smaller than X. But there is work being performed to bring the size of the X Window System down to a sub-megabyte. However, the complexity difference between Microwindows and X will always be great. In embedded-systems designs, there is always the issue of taking maximum advantage of the hardware, since that is what the appliance is being built for in the first place. So when you've got to interface to the hardware for graphics acceleration or when adding a strange touchpad device, it's an order of magnitude simpler implementing this for Microwindows rather than X. And Microwindows' Nano-X protocol is very similar to X, except that it's got a much simpler color model, which makes applications easier to design.

Rick: What other alternatives are you aware of that, shall we say, compete with Microwindows in terms of GUI support for embedded systems and devices?

Greg: I would say that there's probably three competitors in this space, that is, when you're talking about the technology creators. In addition to TrollTech, there's the PocketLinux folks with the Java implementation. Although Century, TrollTech and Transvirtual could be seen as competing, actually each solution is really a different approach to bringing applications to market, and each has its strong points that are well suited in a particular area. For instance, Transvirtual's PocketLinux is a nice Java implementation that runs on framebuffer and implements a small but complete set of applets, which is great if you're running Java. But it won't run any non-java application, so it's not well suited for general-purpose work. TrollTech's Qt/Embedded is in a similar situation but works for a large desktop Linux audience. The Qt/Embedded Project is, along with the entire Qt class library, a fantastic implementation of a very Windows-look-and-feel applications framework that will port with very little, if any, modifications to an embedded device with a framebuffer. However, Qt/Embedded won't run any non-Qt applications. It's also a lot of source code and is considerably complicated.

The Microwindows operating environment is the fastest of the lot and is well suited for general applications development, where it can't be relied on that all the applications will require a certain look-and-feel, widget set or Java implementation. This general environment is well suited for platforms where a large third-party contribution is desirable, such as devices made for the general public or where different technologies must be applied together without restricting the language or APIs used for development.

Rick: Another important requirement in embedded Linux systems with graphical interfaces is the browser. You've started a browser project too—ViewML. What other alternatives are there, and how do they compare with ViewML?

Greg: Of course I'm partial to ViewML, which is the open-source embedded browser created by Century Software. ViewML runs in 800K ROM and 2.1MB RAM, and so it works very well with small devices. But it's quite limited in what you can do, given its small footprint. Recently, a group in Australia ported the full Mozilla browser to Microwindows, and it actually works. So depending on the requirements, people now have both a small footprint as well as a fully featured browser available. Opera gets very high marks for HTML compatibility, and they have an unreleased Microwindows version as well.

With our Microwindows operating environment product, we've been focusing on getting ViewML working well and running fast enough to run on the small PDAs as well as the larger faster ones. In this way, our product comes with a browser that can be used on any device, and we can always upgrade to Mozilla if the customer has enough RAM.

Rick: What were some of the challenges you faced in developing ViewML, and how did you overcome them? What were some of the design tradeoffs that you needed to make?

Greg: Size constraints were the biggest issue, as was the business of making sure that the browser displayed the pages correctly. Since we didn't want to write a browser from scratch, falling prey to the many other browsers that just can't display pages correctly, we selected the KDE Project's kfm KHTML widget. We selected this HTML display widget with two priorities: first, it displayed pages almost perfectly, and second, it was small and well written. The ViewML Project then proceeded to implement an embedded browser without changing a single line of code in the KHTML widget, thus guaranteeing we couldn't screw it up too badly. We wrote a QTÝFLTK class conversion layer, which allowed us to implement all the user interface and widget controls in less than 100K, and the result was ViewML. Most of the work so far has been on finalizing the implementation of this conversion layer. The other consideration was making sure the browser speed is fast enough for everyday users, and we're still working on TrueType font display enhancements in this area.

In the future, we're thinking of moving to KDE's Konquerer v2.0 HTML widget, which features full HTML v4.0 and JavaScript 1.4 capabilities, and that will also be implemented through this conversion layer giving a tremendous decrease in space requirements.

Rick: In retrospect, what has it been like leading an open-source project?

Greg: The Microwindows Project wouldn't be what it is today without the generous contribution of some very important technologies, like the scalable font support, countless debugging on a wide variety of systems and considerable discussion on the mailing list. I must say I've had a heck of a lot of fun being its leader. The Open Source community has had a very positive effect by testing my implementations and others' contributions almost immediately after release. Believe me, if it doesn't work, I'll hear about it.

Rick: Unlike Linux, Microwindows isn't released under the GNU General Public License (GPL). Could you comment on the Microwindows license?

Greg: The early Microwindows contributors and I decided to license Microwindows as MPL, which is a looser license than GPL. This means that Microwindows can be used in projects under NDA or where driver source cannot be made available. Although this is contrary to open-source purists, I'm more of a pragmatist, and my real mission is enabling the growth of the embedded industry with graphical applications. In order to do this, you have to have a system that many people can use, and the license should be as generous as possible.

Rick: What new Microwindows features or enhancements are planned for development over the coming year or so?

Greg: Well, we're going to be producing our freely available binary distribution of the Microwindows operating environment for a lot more PDAs that are on the market or are soon to come to market. This will leverage all developers' interest in Microwindows and allow their applications to run on more platforms.

Another very exciting area will be our inroads into the webpad arena. I've got an architecture designed that will allow developers and users to run the same graphical-applications binaries across a large array of flat-panel-enabled devices, which hopefully will enable the continued fast growth we've been seeing in this embedded space. The applications can take a different look and feel depending on whether the device screen is 240x320 or 800x600, as well as 2-D, 3-D and TV-looking controls being implemented by the toolkits.

Rick: Thank you very much!

[Look for Greg's series of articles on Microwindows in our sister publication *Embedded Linux Journal* in the January/February 2001 issue.]

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com “embedded Linux portal”, which recently became 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.

Welcome to .org Watch!

Leslie Proctor

Issue #83, March 2001

On the Linux Greenhouse, Free Software Foundation Europe, the GIMP and KDE.

The premiere edition of a column is usually the place where the writer introduces him or herself (hi, my name is Leslie, and I'll be your columnist) and stakes out the territory that will be covered (welcome to .org Watch! Our specialty will be tracking the developments and trends in the .org community. We'll profile a .org or discuss a trend or issue and give news briefs). Really, who am I to buck tradition? So, welcome to .org Watch!

For the initiated, what I'm about to say is very, very obvious. However, I'm constantly amazed by the number of people entering into the Linux space who don't understand that the history of the GNU/Linux .org community is the history of Linux. Tracking the trends and developments—of what would be a marginalized group anywhere else—is vital because some of the best innovations come from the .orgs.

Profile—The Linux Greenhouse

The Linux Greenhouse (http://www.linuxgreenhouse.org/) describes itself as a “virtual incubator”. But don't let the word “incubator” scare you. The Linux Greenhouse is a completely different animal from the incubators that have come under fire recently. For starters, it is a nonprofit organization that doesn't charge participants or take an equity stake. In many ways, the Linux Greenhouse is more like a lab than an incubator.

Companies or projects that are chosen to participate must be open source and have high potential. Once chosen, they are given access to resource executives from various Linux businesses. The Linux Greenhouse's mission is to develop a bank of investors and business development, marketing and public relations executives who are willing to donate a little time and expertise to Greenhouse participants. The goal is to give the participating companies access to resources that are hard to come by during the startup phase. It will also give the resource executives an inside track to forming strategic alliances with some small, high-potential companies and projects that may be flying under the radar.

The first Linux Greenhouse class met for a week in Seoul, Korea in June at Global Linux 2000 and consisted of projects and companies from Belgium, China, England, Finland, France, Germany, Korea, Sweden and the United States. Projects include Linux-based intranets in China, a French company that is developing sophisticated Linux-based environments for handheld devices and a web-design system, and a project that is teaching Linux to continuing education students in Sweden. The class of 2001 will be chosen in April, and application forms should be available on-line in mid-February.

News Briefs

Free Software Foundation Europe

Free software advocates on the continent have formed the Free Software Foundation Europe (http://www.fsfeurope.org/), an acknowledged sister organization to the Free Software Foundation (FSF) based in Boston, Massachusetts. MandrakeSoft has donated 2,500 euros to the organization.

“We thank MandrakeSoft for this donation as it will allow us to cover the legal expenses coming up.” Georg C. F. Greve said. “We hope to establish a good and cooperative relationship for future activities on behalf of free software and GNU/Linux.”

The FSF Europe is currently under development and plans to take up work in Germany, France, Sweden and Italy by March. Other European countries, including England, Belgium, The Netherlands and Spain will be added shortly thereafter. The mission of the FSF Europe will be the coordination of free software initiatives throughout Europe.

The GIMP

The GIMP (GNU Image Manipulation Program) is a freely distributed piece of software that can be used as a simple paint program, an expert quality photo-retouching program, an on-line batch processing system, a mass production image renderer, an image format converter and so on. A new stable distribution, version 1.2, has been released and can be downloaded at http://www.gimp.org/.

The GIMP recently won the Linux Community Award for “Most Innovative Desktop Software” at the Systems 2000 show in Munich. Carey Banks, Marc

Lehmann and Simon Budig accepted the prize, which included a trophy and a check for 3,000 DEM.

KDE

KDE (www.kde.org) has released XParts, an extension written by Matthias Ettrichm, Simon Hausmann and Lars Knoll, which extends KPart and makes it possible to embed out-of-process components. XParts provides interoperability with major UNIX/Linux software toolkits and applications, including Mozilla. It is now possible to use Mozilla's rendering engine (Gecko) instead of KHTML in Konqueror. This can be set up at runtime in a config dialog and requires no changes to Konqueror.

Eventually XParts will allow Linux and UNIX developers to make KDE components—no matter which toolkits and software they decide to utilize.

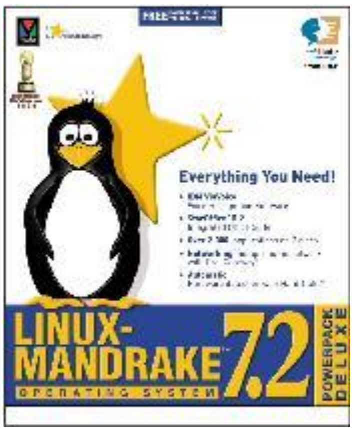
Leslie Proctor is an active participant in the .org community and has acted as a consultant for a number of .org organizations around the world. Send mail to lesproctor@yahoo.com.

Mandrake 7.2: Odyssey to Mediocrity

Stephanie Black

Issue #83, March 2001

Stephanie takes a look at Mandrake 7.2.



- Manufacturer: Mandrakesoft
- E-mail: services@mandrakesoft.com
- URL: http://www.linux-mandrake.com/
- Price: $69.95 US
- Reviewer: Stephanie Black

Numerous Linux distributions have issued new releases in the last few months. Most of these have a target group of users (or, at least, a target group of competitors). It's of necessity a shock when, laden with packages of every description, a rather graphically intensive version of Linux appears with so many wonderful ideas and such incomplete implementations; one is left reeling. It's rather like a pot-luck, all brought by one individual, with such a vast array of dishes that you tend to feel a mite guilty for thinking the individual in question might have cooked the chicken for another 30 minutes. You don't want to seem ungrateful, but there's the small matter of salmonella just waiting for you.

For review purposes, Mandrake burned a prerelease set of disks. These contained what was ostensibly to be on the published CDs. There may have been changes.

Installation

Graphical installations can be taken seriously by the more technically minded, contrary to popular mores. I can install and configure from the command line with the best of them, but I'm on the hunt for a decent installation package that neither threatens my mere 64MB RAM with KDE nor is reminiscent of nursery décor. The hunt continues.

Even the most advanced Linux users among us occasionally make mistakes, so the absence of a back button (at any point in the installation) is on the “Most Glaring Errors” list. This is a distribution that is targeted to, along with pretty much everyone else, new Linux users. New users are a demographic that tend to make errors or at least typos. Back buttons mean not having to interrupt your install to reboot and start over.

That said, there are some lovely additions to the install process. Configuration of networks, most hardware and the system itself is made incredibly simple and, for those of us who know how to do this manually, faster.

Package Overview

In spite of the choice to select individual packages or package groups, this process is daunting; 2,300 applications is a fair amount of toys to play with, and a number of the more notable ones were not able to be included here. Open Sales AllCommerceTM E-Commerce Tool, Broadcast Video Editor and IBM's Via Voice were among those not used for this review for reasons of system configuration.

The documentation available for Mandrake 7.2 is extensive; in addition to the usual HOWTOs, man pages and such, there are some commercial books: *Practical Linux* (Que, 2000) and *Linux Hardware Handbook* (Sams, 2000) are the more indispensable volumes included.

One rather nasty gaffe in the documentation was in the link to Mandrake's English reference from Netscape, which should read:

```
file://usr/share/doc/mandrake/en/mdkrefguide/index.html
```

but doesn't.

There is an update utility that obtains bug fixes, new versions of packages and the like. It's a sort of “apt-get upgrade” for Mandrake, all wrapped in a GUI. It may make the difference between liking and loathing your system.

The winning application (that is, the one that did not crash, burn or otherwise suffer from extended running, testing and learner-abuse) was ForteTM for Java. Granted, this is Sun's brainchild, but that it runs under Mandrake rather well is a good sign. Even IDLE didn't manage to initialize completely and then refused to do anything during subsequent attempts to start it.

The absence of screenshots here is indicative of what can go wrong under Mandrake 7.2. I've had no problem with either the GNOME screenshooter applet or the GIMP in any distribution—until Mandrake. More's the pity, as the GUI really did look quite nice on one or two occasions, and it would have been advantageous to have pictures of the configuration tools. A good many of the standard Helix GNOME applications and applets malfunction or cease functioning in Mandrake. This, at one point, included the gdm (which ironically, was the acronym for my response to yet another “oops” from Mandrake).

Appearance

The 4.0.1 version of X is included with Mandrake as the default. XFree 3.3.6 ships as well for those who don't need the extra functionality of 4.0.1. As stated earlier, this is configurable with neither extensive experience or time requirements.

I enjoyed the full range of themes available for Sawfish, many of which are missing under other distributions. What other distributions tend to have, however, is at most one or two backgrounds that could be considered “self-promoting”. Additionally, they don't give the impression of being designed for a four-year-old's bedroom.

Conclusion

It's anyone's guess whether runtime problems are due to libraries, or to unflagged dependencies, even the “optimized” Mandrake kernel 2.2.17. (This is an issue I'll discuss another time, but suffice it to say that an increasing number of Linux distributions have souped-up kernels and are inadvertently making an argument for standardization in the process.) For whatever reason, or combination of reasons, a good many of the applications included with Mandrake are not able to run or don't run as smoothly as they do under other distributions.

Mandrake is the kind of distribution one *wants* to like very much. It's cheery, and the good folks at Mandrakesoft have been so generous with packages, one wants to be able to say, “Well done!” The best one can, honestly, get away with is to admit that it's good in some parts. Most frequently, those parts have to do with ideas, rather than the implementation thereof.

Good/Bad

Stephanie Black is a writer—of words and code. When not writing, she runs a Linux consultancy, Coastal Den Computing, in Vancouver, BC, Canada. In her off-hours, she's usually playing fetch with her cats, or collaborating/colluding with her partner, a fabric artist and business manager.

Archive Index Issue Table of Contents

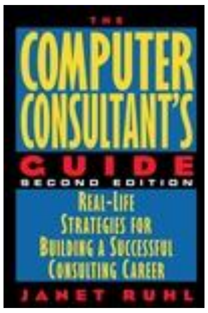
Advanced search

The Computer Consultant's Guide, 2nd Edition

Ralph Krause

Issue #83, March 2001

If you're considering becoming a consultant, or have been consulting for a while, The Computer Consultant's Guide can help you understand which skills and practices contribute to being successful


THE
COMPUTER CONSULTANT'S
GUIDE
SECOND EDITION
REAL-LIFE STRATEGIES FOR BUILDING A SUCCESSFUL CONSULTING CAREER
JANET RUHL

- Author: Janet Ruhl
- Publisher: John Wiley & Sons, Inc.
- URL: http://www.wiley.com/
- Price: $24.95 US
- ISBN: 0-471-17649-4
- Reviewer: Ralph Krause

Making a living as an independent computer consultant is not an easy task, and running a business requires skills that technical people might not be comfortable using. If you're considering becoming a consultant, or have been consulting for a while, *The Computer Consultant's Guide* can help you understand which skills and practices contribute to being successful.

Written by Janet Ruhl, a leader of CompuServe's Computer Consultant's Forum and a regular contributor to “Contract Professional”, this edition of the book has been updated and now includes information on consulting in Canada and Europe. The book contains 290 pages divided into eleven chapters, appendices and an index.

Subtitled “Real-Life Strategies for Building a Successful Consulting Career”, *The Computer Consultant's Guide* covers all phases of a consulting career from determining if you're ready to start consulting through increasing your business once you are successful. The book contains quotations from many consultants, which demonstrate real-life examples of the ideas presented.

The first chapter of the book provides a general explanation of what different computer consultants do. It covers contract programmers, resellers, custom application developers, multimedia/web authors and management consultants. For each type of consultant there is a “Meet the Consultant” section that contains thoughts from a consultant practicing in that field. The chapter ends by providing common questions and answers about computer consulting.

The next chapter helps you to determine if you are ready to begin a consulting career. It is important to examine both your skills and your pool of potential clients. It explains that finding the first client is usually easy, but that finding enough of them to make a living is difficult, so there is advice on how to build a pool of potential clients before striking out on your own. It also offers advice on personality and business skills that will help you to be a good consultant. Potential financial pitfalls, such as the loss of insurance and difficulty obtaining loans, are explained. The chapter ends with a discussion on preparing your family for your new job and contains specific information for mothers who are wondering if consulting will let them stay close to their children while earning a good income.

Chapters Three through Seven deal with the mechanics of running a consulting business. Covered topics include obtaining licenses, deciding if you need to incorporate, creating a business image, marketing your services and working with brokers. Most useful for beginners are the chapters on determining rates and successful marketing strategies. Chapter Five addresses the problem of determining what to charge and helps you decide if you should ever work for nothing. Chapter Six covers marketing and stresses that word of mouth is your best type of marketing. It does offer advice on mass mailings, brochures, advertising and cold-calling if you feel that you must try these. Chapter Seven gives advice on working through brokers, especially on how to find a reputable one and how to avoid Draconian noncompetition clauses that can prevent you from working with old clients once you leave the broker.

The next three chapters cover client relations—from interviews with potential clients through collecting payments. Chapter Eight helps you conduct an interview with a potential client, determine when to start charging them and explains common contract clauses of which to be wary. Chapter Nine concerns consultant behavior while working for a client and includes advice on dealing with common problems, both technical and personal, encountered at a client's site. Chapter Ten deals with support and payment issues. It shows how support for old clients can be profitable and provides advice on how to support them without slighting new clients. It also discusses what to do if clients are slow to pay, won't pay or can't pay.

The final chapter business, dealing with success and avoiding burnout. If you have more work than you can handle this chapter helps you decide between hiring more employees to handle the load or taking on less work. If your business has plateaued, it explains how you can increase business by doing such things as selling hardware, giving seminars and/or writing. It also helps you avoid getting in a rut from doing the same type of work year after year. There are times when leaving consulting and reentering the corporate world will appeal to you, and this chapter helps you determine when to do this and provides information on doing it gracefully.

The book contains two appendices and an index. Appendix A is a list of resources for consultants, including print material, professional organizations and web sites. Appendix B lists the twenty factors that the IRS uses to establish your status as either a contractor (consultant) or an employee for clients.

The book is easy to read, and Ruhl's extensive experience in dealing with these issues is apparent. This book was published in 1997, but it is not outdated because it focuses on business skills, not technical ones.

The Computer Consultant's Guide provides easy to understand, practical information for consultants just starting out or for people considering entering the field. If you've tried consulting previously but were unsuccessful, it can help you determine what went wrong. In addition to helping you determine when you're ready to start consulting, it can also help you avoid common consulting pitfalls and keep your business growing.

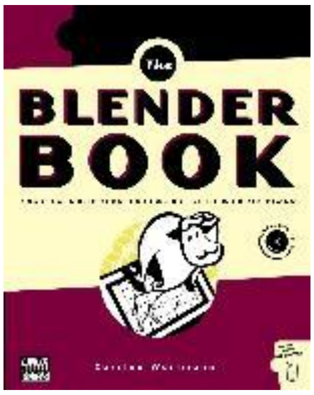
Ralph Krause lives in southeastern lower Michigan and has been using Linux for over two years. His goal is to be able to make a living using Linux. He can be reached at rkrause@netperson.net.

The Blender Book

Clifford Anderson

Issue #83, March 2001

Blender is a free, Linux-based 3-D program, that has gained a fairly rich following with Linux users.



- Author: Carsten Wartmann
- Publisher: No Starch Press
- URL: http://www.nostarch.com/
- Price: $39.95 US
- ISBN: 1-886411-44-1
- Reviewer: Clifford Anderson

In the world of digital imaging, there are two classes: 2-D and 3-D. *The Blender Book* by Carsten Wartmann deals with the latter. Blender is a free, Linux-based 3-D program, that has gained a fairly rich following with Linux users. Wartmann's book is currently unique in that it is the only book to date dealing with the topic of teaching Blender to the rest of us. Therefore, it would be unfair to class his book as “best of breed” alongside other books available on the same topic, nevertheless it warrants a good review by virtue of its strength of content.

That said, the book itself sets a path to take the reader on a topical, hands-on journey through the various aspects of Blender's features. “Features” means two things here: those that are specific to Blender as a program, and those that are specific to 3-D modeling. This is to give fair warning to the potential reader. Wartmann is not shy about displaying his technical literacy regarding the details of the 3-D world (i.e., he doesn't water down the language to make the book more palatable to the casual reader). What he has set forth in writing is a willingness to give us a straight, no-nonsense approach for understanding 3-D in general, and then to go about the business of using Blender to render our work.

With regard to the book itself, one of the more virtuous aspects is its self-containment. The book comes with a CD-ROM that has the latest version of Blender (version 1.8 upon publication), a copy of Python (for scripting) and a substantial number of files that the reader may bring into Blender and view in all their detail. What is more, Wartmann has included all the necessary graphics to compose the lessons for each chapter and has rendered all the motion 3-D lessons as *.avi movies for the reader to capture the essence of each lesson before getting started. Very nice, indeed.

The content of the book does not assume that the reader has a knowledge of 3-D, so the author begins with explaining the world of 3-D rendering by covering the basics that will be necessary for doing the tutorials throughout the remainder of the book. What I found most intriguing is Wartmann's ability to describe the huge arena of 3-D and reduce it in manageable chunks so that readers get sufficient detail, all the while not finding themselves bogged down with information overkill. Wartmann, therefore, is consistent with his intent for writing his material: to give the reader an adequate understanding of both 3-D rendering and the use of Blender in “a fast, practical, and compact introduction to the world of 3-D graphics and animation...” (page 2)

Those who do have a knowledge of 3-D rendering will appreciate the book for its hands-on approach to understanding the interface and functionality of Blender as a program. Blender is not Maya or 3-D Studio Max. It is a coming-of-age program that is continuing to be developed under the company, Not a Number. That said, Blender most assuredly has its own way of creating 3-D graphics, in that its interface is quite different from that which other, more commercially prolific, 3-D programs offer as standard. Wartmann shines as an author because he keeps the knowledgeable student interacting with Blender's unique interface by virtue of keeping his detailed analysis of 3-D rendering specific to Blender, as well as keeping us abreast of where we are in the program with plenty of screenshots in order not to lose sight of the program's interface while following his prescriptions.

Without exhausting the book's content here, it can be said that Wartmann does a good job at covering the basics for 3-D modeling and rendering. From understanding simple shapes (bezier curves, NURBS, text and metaballs) to creating realistic animations via Inverse Kinematics (IK) and radiosity, the author breaks these concepts down into chapters and subsections, conquering each topic with a hands-on tutorial. Once we get past the introduction to 3-D and Blender's interface and terminology, our task for interacting with the book is create, create, create.

For example, Chapter Five starts with creating primitives (simple shapes: spheres, cylinders, etc.), moves us forward by understanding the math behind it and how to implement the math to fine-tune a primitive via the controls of Blender's interface, then adds the issues of rotating, smoothing, extruding and warping a given primitive. Once those basics are covered, it's on to creating a mountain terrain, or landscape. Add to this a texture fill and a little tweaking with curve (bezier) editing, and we have a finished project before us. What is most effective about Wartmann's approach is, while doing the tutorial, he is constantly revisiting and/or adding 3-D concepts to our learning curve while introducing the tools at hand. This helps the reader to stay in the world of 3-D modeling and all its aspects without sacrificing the forward movement of creating the content itself.

Here's a specific example of what the reader can expect from Wartmann's book. NURBS are as common in 3-D rendering as is a line in vector illustration. But the author makes no assumptions. Therefore, when introducing the use of NURBS, he shows us his cards straight up: “NURBS (Non Uniform Rational B-Spline) curves are three-dimensional curves represented by a class of equations defined by nonuniform parametric ratios of B-Spline polynomials. (That's probably more than you wanted to know, but there you have it.)” (page 83). Right. From there, he moves on to how this is translated via Blender's interface and then moves us into creating a simple NURB curve. The amount of material to cover all this? Two pages. The point to be made here is, although Wartmann is terse in some of his explanations, it is not at the sacrifice of the reader's need to understand the context by which he is creating 3-D images/ animation from within Blender. The author covers all his bases within a reasonably short amount of space.

The last sentence above is one of the very few issues I have with the book itself. It's too small! 3-D is a healthy topic all by itself and using Blender is no small feat. At first blush, opening the program is to become a little overwhelmed by its sheer girth of functionality. Although Wartmann concedes that his book does not exhaust every aspect of Blender's power, I certainly would have liked him to give it a try! Nevertheless, what he does do is give the reader a sufficient amount of detail to have us up and running with the basic concepts of 3-D rendering from within Blender in both its complexity and flexibility.

Those who are new to 3-D and want an excellent (and free!) program by which to indulge their curiosity will appreciate having this book to get them quickly up to speed with Blender. Others, who are already familiar with the concepts of 3-D imaging, will not be disappointed by the approach of Wartmann's book because its main concern is to familiarize the reader with using the unique angle with which Blender creates 3-D images/animation.

Clifford Anderson is a freelance writer and graphic designer. Currently residing in Portland, Oregon, he can be reached at aclifford@uswest.net.

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #83, March 2001

Readers sound off.

Offended

I notice your December 2000 cover shows an IT guy as a fat, balding, bespectacled, bearded geek in an XXL black T-shirt and with an attitude problem. I'm most insulted by this stereotype. I shave.

—Bruce Richardson brichardson@lineone.net

Our cover model, Glenn, responded to an open ad to system administrators and was chosen for his expressive qualities.

—Editor

Lose the Windows

I recently received the December 2000 issue of *Linux Journal* and went right to one of your feature stories—“A Linux-Based Automatic Backup System”. I was expecting to see something about tape drives and backing up and restoring a dead hard drive. Instead, I see a very brief tutorial on using Samba, cp, cron, and bzip on backing up Windows shares on a Linux hard drive. The author quotes that it's an “inexpensive and easy solution”—he's exactly right about that. I don't see creating a zip file on a hard drive a backup solution, nor do I see this article being a feature story. There was absolutely nothing in the article that even mentioned backing up Linux files or how to restore them. Articles like this and the wave of many others that you have published in the past are a complete embarrassment to Linux. They all seem to be Linux acting in a supporting role to Windows. Come on! Look at the front cover of your magazine! Publish some real articles about Linux for a change!

—Paul Sullivan wiley14@mindspring.com

A Real Bastard

Thank you for your short article in the December 2000 issue of *LJ* on the trouble with the Bastard Operator from Hell. Interesting from my point of view that I've never felt that my job has sucked. It's often busy and I have had to deal with a lot of bozos, but I've generally been having a good time. And I'm somewhat in the minority in that I did major in computer science.

I do have one question, though. I've just recently (this week) started at a new employer. Problem is, I've run into someone in my group who likely doesn't think BOFH is funny at all. Rather, he thinks it's a manual on how to be a systems administrator. I kid you not. So what do you do when you encounter a real life BOFH? It's serious enough that I think the guy should be just outright fired. I still don't know how he got hired and how he's lasted as long as he has.

—Anonymous

Unfortunately, there's no Getting People Fired HOWTO. However, Simon Travaglia's BOFH stories, The Art of War by Sun Tsu and *The Prince* by Nicolo Machiavelli are all available on-line.

—Editor

Common Misconception

In Reuven Lerner's “At the Forge” article in the December 2000 issue he states “...CGI has a number of drawbacks...,it requires that the web server spawn a new process.” It saddens me to see that a regular columnist at *Linux Journal* misunderstands the meaning of CGI.

Even though he defines CGI as “common gateway interface” he fails to continue with the thought. It is an interface to a server. It is not a program. Any process that processes data given to a server with a QUERY_STRING is a “CGI program”. That means that mod_perl scripts, PHP, servlets, JSP, etc., all of which run in the server memory space—thereby not spawning new processes—are also CGI programs.

Yes, the common misconception is that CGI is a type of scripting, but as the leaders of the industry we should be precise in our language and meaning.

—Nathan Hokanson nathan.hokanson@home.com

Ada Boy!

I enjoyed the article by Frode Tennebo on Ada 95 programming in the December 2000 issue. It's exciting to see Ada's increasing popularity in the Linux world. It's because of the language's features and robustness that best-selling Linux author Warren W. Gay announced that he would be porting his open-source projects to Ada.

For the benefit of your readers, more information about Ada Linux programming is available through the *Big Online Book of Linux Ada Programming* (www.vaxxine.com/pegasoft/homes/book.html). This is one of the largest on-line resources for Linux programming for any language and is a good starting point for newbies.

Thanks for letting your readers know that there are alternatives out there to C++ and Java.

—Ken O. Burtch kburtch@sympatico.ca

Wacky Names

As a subscriber for about a year now, I wanted to say that I find the magazine very helpful and informative. I enjoy the section that displays wacky domain names that have not been consumed by the masses yet. I happened to have stumbled across a web site name that helps finding these names (http://www.namedroppers.com/). Enter one (or more) keywords and it will display *all* registered domain names that contain the words. Of course, I did a bit of scouring and yes, they are running Red Hat Linux. Just a tip and keep up the good work.

—Jim V. jvanar1@uic.edu

Penultimate Linux Box?

I loved the article in the November 2000 issue titled “Build the Ultimate Linux Box” but you stopped short of actually delivering on the promise of the article! Sure, there are lots of choices out there for a motherboard, hard disk drive, etc., but knowing that doesn't help in trying to assemble “The Ultimate Linux Box!” Knowing that Tyan, ASUS and Abit make good motherboards is like answering the question “Why Linux?” with “Because it does lots of cool stuff!” I hope that you will do an article similar to the one at www.maximumpc.com/reprint/archives/archive21.html but focused on Linux. I think it's great to give the pros and cons and considerations of each component, but then take a stand! Commit yourself to something and tell us why! That is a lot more useful than a wishy-washy “Well, this is cool, and that's cool, and these other folks say that all these other things are cool too, so they would all be good in an Ultimate Linux Box.”

—Jerel Crosland Jerel.Crosland@21stcenturyinsurance.com

SuSe Too Loosa

In regard to the SuSE 7.0 review [*January 2001*], I used SuSE 6.1 and 6.2. I tried to upgrade to 6.4 and it kept reverting to German with a German keyboard. I could fix it at the command prompt and KDE bug team said this was a problem with SuSE. It's a good program and if one speaks German it has support. The Cal crew [*California office*] is clueless. I took it off the machine. I need the newer version to play with Star Office and its database. I think the basic problem with the Linux crowd is that no one gives a rat's a** about the end user.

—Paul Taylor ptay1@bestweb.net

I'm sorry to hear you had such a problem with SuSE. I'm not a SuSE employee, but I will say my experience with 6.4 and 7.0 has been pretty good. Definitely no issues with German keymaps. You mentioned upgrading to 6.4. My habit, with any distribution, has been to avoid upgrades and do a clean install. Linux is changing too rapidly, and I think you are asking for trouble doing an upgrade. I generally back up everything (you do have a backup strategy and media right?), repartition (possibly), reformat and do a clean install, then pull back my personal files (e-mail, bookmarks etc.) from the backup. The newer distributions seem to pick up most of your hardware right out the gate, so a lot of that hassle is now gone. Any remaining favorites I may be missing I'll reinstall from the distribution's CDs, rebuild from source, or restore from my backup. This may seem like a bit more hassle than an upgrade, but in the end I think you'll end up with a much more stable system. If all you're after is StarOffice, you can just add it to your current setup. I've done this on two Mandrake 6 systems with no problem, downloading it from Sun.

—Stew Benedict stewb@aysenterprises.com

LJ Interactive

I want you to know that interactive.linuxjournal.com is one of the best features any magazine could have for their publication. There are countless times when I can't remember the name of an article or when it was, but I remember the subject. The ability to have all the articles on-line and searchable is a godsend; I wish more magazines, especially technical ones, would do something similar.

—Sheldon Dubrowin dubrowin@yahoo.com

For better or for worse, we're one of a kind.

—Editor

Sold on *Soldier*

Loved the review on *Soldier of Fortune*. Yep, I'm sold. I want this game. I'm ready to experience the adventures of mercenary John Mullins; this is something—definitely “Games Penguins Play”. I must say the cover to *Linux Journal* January 2001 was fabulous. King Darius was a real eye-catcher, I hope you will do more covers in this fashion!

—Paul Dale Roberts Silhouet98@cs.com

groff is Great

I am writing to thank you for publishing Wayne Marshall's article on groff [December 2000]. Even though I have been using groff extensively for four years, I still found something new to learn in Wayne's piece.

I learned about groff some years before when I obtained a copy of *UNIX Text Processing* by Dougherty and O'Reilly. I found it intriguing and poked away with the **ms** macros for a while until I returned to school and decided that the courses for which I was paying deserved to have my handwritten (and largely illegible) notes transcribed for future reference.

Wishing to “give back” to the Linux community, I prepared a “Groff and Friends HOWTO” (http://www.ucalgary.ca/~dprovins/), which after reading Wayne's article might be more aptly titled a “groff and Friends ms HOWTO”, as it is focussed on that set of macros, and how to adjust them to meet one's needs.

Even with all that “experience”, I still found Wayne's article enlightening, and when it comes time to rewrite the how-to (which it could use), I hope to include some of his insights. Naturally I'll mention his article as a reference for new or not-so-new users.

May I add that there is an active mail list for groff at http://groff.ffii.org/ that I recommend for those with both questions about, and suggestions for, groff development.

—Dean Provins provinsd@ve6cta.cuug.ab.ca

What's up with Ogg?

In my January 2001 copy of *Linux Journal*, there is an article on Ogg Vorbis as an open-standard alternative to MP3 ["Ogg Vorbis-Open, Free Audio—Set Your Media Free"] The article states that Ogg Vorbis' main financial backer is iCast, an entertainment division of CMGion. However, it makes no mention of the fact that CMGion closed down iCast services as of the end of last year and is closing many other divisions of its company acquisitions.

I was just curious why such an article was run with information that was old news or at least did not include an update when it seems that the backbone of this standards support is suffering from osteoporosis.

I checked both the Vorbis site and the xiph site linked from the article, but neither of them have any news or apparent updates since mid November (when icast was shut down according to ZDNet news).

—Kathy Lynn

Kathy, As you say, iCast services were closed down as of the end of last year. The issue of the article is January 2001, meaning it was on the streets mid-December and at the printer in November. The article's author, Jack Moffitt, former VP of technology for iCast was laid off when it happened and had no idea at the time he wrote the article. There is, however, already much software that supports Ogg Vorbis available at www.vorbis.com/software.htm.

—Editor

Changes to the *Python Developer's Handbook*

I just received my February 2001 issue of *Linux Journal*, and it was great to see a review of my book there. I liked every single word that you said.

However, it seems that the guys from SAMS didn't tell you about the last-minute changes in the book. On October 16, when version 2.0 was released, I called my editors to let them know about it, and they asked me to cover version 2.0 in the book too. Therefore, the book that went to the printer also has an Appendix D "Migrating to Python 2.0". Besides that, I added a lot of information throughout the book concerning version 2.0.

—Andre Lessa

Errata

Frank LaMonica's "Streaming Media" (January 2001)

The calculation of the bandwidth 8000 clients (80% of 10,000) produce is not 30GBs but 300MBs. This amount can be handled by today's possibilities in contrast to the published number. The second error is in the RAID naming. RAID 0 is striping, which is increasing the size (and speed) of a volume by adding them together in a row. The RAID level Frank describes is RAID 1 (mirroring).

Mick Bauer's “Paranoid Penguin” (January 2001)

Bauer incorrectly refers to Tatu Yloenen as being associated with F-Secure; in fact, he's chairman of the board and chief technical officer of SSH Communications Security, Inc. Mick also wishes he had credited at least Markus Friedel, Niels Provos, Bob Beck and Aaron Campbell. He would like to correct his statement that SSH Communication's version of SSH v.2.3 must be purchased if used in a commercial setting. That is not true for users of open-source operating systems:

> To qualify for a Non-Commercial Use License, You must: (1) use the Software solely on a system under the Linux, FreeBSD, NetBSD, or OpenBSD operating system (whether for commercial or noncommercial use);... [SSH Communications Security Corp SSH(R) Secure ShellTM License Agreement, paragraph 3]

—*Editor*

UpFront

Various

Issue #83, March 2001

Stop the Presses, *LJ* Index and more.

LJ Bashed in *WSJ*

It was P.T. Barnum who famously said, “I don't care what you say about me, but just spell my name right.”

So that's our rationale, here at *Linux Journal*, for enduring a bit of publicity that came our way via the December 14, 2000 issue of *The Wall Street Journal*. The “Digits” column on page B6 (the Technology Journal page) leads off with a six-inch item titled “Linux Battles”. Normally we scan pubs like the *Journal* for tidbits about anything and everything that might remotely relate to Linux. But this time the news wasn't just close to home—it was home. The story was about *Linux Journal*.

“Digits” is the *WSJ*'s form of “UpFRONT” and shares the same appetite for irony. Unfortunately, the irony in question here involved the apparent fact that our modest little on-line store sold police-style barricade tape that says “Microsoft Free Zone” while the company that actually runs the store, WAS, Inc., was hardly Microsoft-free. It seems that the site was at least partly served up by (shudder) Microsoft Windows NT—at least while WAS moved its operation onto some kind of UNIX.

We have been working with WAS to hasten the end of this irony and expect to return the store to an equally Microsoft- and news-free condition.

Storage System Turns Latino Research Center Into Publishing Powerhouse

Founded in 1989 at Michigan State University, the Julian Samora Research Institute has one purpose: to generate, transmit and apply knowledge that will serve the needs of Latino communities in the Midwest. Grants to the Institute help to fund empirical research done by scholars and to publish their research as books or monographs. This research looks at relevant social, economic, educational and political conditions of Latino communities in both the US as a whole and the Midwest in particular. The Institute's forthcoming data will serve as a resource on and for Latinos.

The Institute started publishing small books and reports a decade ago. Since then, it has increased both its research and publishing volume ten times over. Up until three years ago, the Institute could get away with publishing a book on paper and then file it away.

Danny Layne, who divides his time between network administration and publishing production, says: “If we needed to print a book, we'd pull it out of the file and then put it away.”

To keep up with the volume of research it had to publish, the Institute found itself producing more electronic files—files that kept getting larger and more complex. Researchers broke down chapters into multiple files. One book could consist of 20 different files. Researchers also generated electronic charts and graphics along with PowerPoint presentations. Books got published not only in hard-copy format but also on-line on the Institute's web site. Layne says: “To this end, we were generating new types of files that we never had before.” Disk space on a desktop personal computer couldn't handle the volume being churned out. Layne says that the Institute didn't want to start adding large hard disk drives to its desktop PCs. “If one PC's disk drive failed, then we'd have to restore files from a previous backup tape and recreate what we lost. That's inefficient.” So, with technology funds from the government and the University, the Institute decided to buy a central storage system to house all publications and the files for the web site. Since the Institute has a small computing staff and limited resources, the storage device had to be highly reliable, easy to set up and maintain and able to accommodate more storage space with the addition of more disk drives as needed. Layne notes, “Our search for a storage system brought us to Winchester Systems. We purchased a FlashDisk external RAID storage system with seven 9GB disk drives.”

Layne observes that just three years ago the Institute had virtually no storage—only a few desktop PCs. “During this time, the FlashDisk has allowed a small research department, within a large university, to turn itself into a publishing powerhouse on a small purse. Some of the other departments on campus are in awe of our storage system. And there are good reasons for it.” Two side-by-side Dell PowerEdge 2200s, one Windows NT and one Linux plug directly into the FlashDisk. It provides fast, highly reliable RAID 5 storage to multiple servers with different operating systems. This feature eliminates the expense of buying storage for each server. Layne says that managing one storage system is easier than managing two or three of them.

The Windows NT server, which connects to the intranet within the building, functions as a central repository for all active publications and for the databases used to inventory and to track these publications. The FlashDisk allows each researcher to have his or her own storage space, apart from the desktop. Using either a Windows-based PC or a Mac, researchers can access both Windows NT, Mac and Linux files stored on the FlashDisk.

Cross-platform programs allow the system to function as a quasi-network attached storage filer. These programs include Services for Macintosh running on the Windows NT server and Netatalk running on the Linux server. The latter program permits printers to function as network devices.

The FlashDisk also contains a large collection of artwork. Overall, the FlashDisk provides the researchers with fast access to a large bank of files: everything from text to graphics, regardless of the format, over an intranet. When a book is no longer going to be published, it gets archived to a CD-ROM or a DVD. Meanwhile, the Linux server, which connects to the external network, contains all of the Institute's web files, as well as the web site itself. About 700 web pages reside on the FlashDisk. The web site gets about 3,000 hits each day (100,000 hits a month). Setting the space aside on the FlashDisk to store the Linux files, as well as the Linux operating system, turned out to be easier than Layne thought it would be: “We just followed the FlashDisk's instructions in the manual and made one telephone call to technical support, and then we were up and running,” he says.

While the FlashDisk provides a large amount of disk space, Layne wants to avoid having it become clogged with multiple versions of old files. He says that keeping storage neat and trim shouldn't become a time-consuming burden for a network administrator. “We've mentored our researchers to perform a number of storage housekeeping procedures. After all, they're responsible for overseeing their flow of information, including creating it, updating it, storing it, deleting it or archiving it.” For example, researchers learn how to name their files so they can easily locate them and remove them if they get old. Layne has also put a regular storage clean-up program in place. Researchers have to go through their storage space and either delete multiple copies of files or move old files to a CD. While researchers do a good job of maintaining their space, Layne says that the Institute's publishing volume has a healthy appetite for more storage space.

According to Layne, “We're planning to upgrade our 9GB drives to 18GB drives to double our amount of storage. We can do this inexpensively because

Winchester Systems will give us credit toward a trade-in on the drives. The service folks at Winchester Systems must feel like the repair people at Maytag. The FlashDisk has never broken down, not even hiccuped.”

Money from the University will allow the Institute to produce audio and video clips for the Web. Layne says, “We've already tested accessing and storing multimedia on the FlashDisk. Everything worked fine.”

—Elizabeth M. Ferrarini

Job Opening Trends

by Reginald Charney

Fuzzy Data

There was a plethora of terms found when analyzing the job-opening descriptions. For example, Windows' terms include 95, NT, CE, Win, Windows NT, etc. Besides various abbreviations, there are also ambiguous words. Is Win followed by NT to be counted as Windows and Windows NT or just Windows NT alone? Generally, the “maximum munch” rule is used. That is, the longest recognizable term possible is used. Then there are still the misspellings and unknown words. Thus, all this makes determining what is required fuzzy. Having said that, here are some of the statistics used in the job descriptions:

- number of job openings: 129,000
- number of unique words: 12,200
- number of unique skill sets: 68,800

Five most often used words:

- C++: 19,951
- Java: 16,920
- SQL: 9,121
- vb: 9,003
- HTML: 8,788
- JavaScript: 4,793

As you would expect with all these terms, skill sets made up of a couple of words would tend to appear more often than skill sets made up of three or more words. The top five skill sets were:

- frame relay: 445
- Oracle dba: 445

- C++ Java: 412
- WinNT server: 361
- Shell script: 361

Thanks to DICE at www.dice.com for a truly valuable service and allowing me to analyze some of their data.

> **Reginald Charney** currently heads the US chapter of the Association of the C and C++ Users. Visit their site at http://www.accu-usa.org/ to learn more.

Linus Torvalds, Then and Now

Linus Torvalds [from *LJ* March 1995]: “I'll make a [kernel] 2.0 someday...” “No wonder Linux works so well. He has an alpha-testing lab of ten thousand people!” says someone from Novell.

Mr. Torvalds [from *LJ* November 1999], married with children, well into the 2.x series of kernels with eight million users strong.

LJ Index—March 2001

1. Market cap in billions of USD at which Yahoo is worth more than all magazines put together: 30
2. Number of billions of USD IBM is investing in Linux software, hardware, services, partnerships and the Open Source community in 2001: 1

3. Number of servers X-series computers in what IBM claims will be the world's largest Linux supercomputer: 1,024
4. Number of racks used to house all those Linux boxes: 32
5. Sum in trillions of USD that will be spent on Internet infrastructure and e-commerce by 2003, exceeding the gross domestic products of Germany, France and the UK: 2.8
6. Year by which Vint Cerf expects the number of net-connected devices will outnumber the world's telephones: 2006
7. Millions of net-connected devices, independent of cell phones, Vint Cerf predicts will exist by 2006: 900
8. Number of airline flights in the next 24 hours: 42,300
9. Number of people who will take those flights: 3, 000,000
10. Number of flights that will crash: 0
11. Number of travelers in 1999 who used the Net to plan trips and make reservations: 52,000,000
12. Percent of application developers worldwide who say they plan to write wireless applications in the next year: 40
13. Position of Linux as a web server platform: #1
14. Percentage of web servers that ran on Linux as of May, 2000: 36
15. Position of Linux among fastest-growing server operating systems: #1
16. Internet-related applications as a percentage of spending on Linux servers: 40
17. Number of handheld and notebook information appliances by 2002: 55,000,000
18. Year by which shipments of handhelds and notebook appliances will exceed shipments of PCs: 2005

Sources:

- 1: *Forbes*
- 2-4: IBM
- 5: Nortel & International Data Corp. (IDC)
- 6-7: Domain Street
- 8-10: Boeing
- 11-12: Evans Data Corp.
- 13-14: Netcraft
- 15-18: IDC

Linux Bytes Other Markets—Sticky Game Site Powered by Apache and Linux

After only a few months of operation, NeoPets.com, a web site built on a Red Hat Linux/Apache platform, is already turning a profit, recording billions of page views monthly. Targeting youths aged 20 or younger, the site enables users to create and care for their own personal virtual critter known as a “NeoPet”. It also boasts a series of constantly changing “universes” complete with games, stories, contests and entertainment. According to recent figures from PC Data Online, NeoPets attracts 2.1 billion page views and 2.3 million unique users each month who each stay for an average of 7.48 hours, making this the stickiest site on the Web.

Based on August numbers, NeoPets ranks higher in page views than Excite, Lycos and Amazon. What's more, it engenders far more loyalty (termed stickiness) among users. The average AOL user, for example, visits for 35 minutes a month, while Yahoo users spend three hours 22 minutes. In the Gen Y market, NeoPets total of seven hours 48 minutes trounces the competition, with ten times the page views of Disney.

Initially created in a college dorm with a “launch campaign” that consisted of sending a couple of e-mails to other virtual pet sites, the site chalked up 200 sign-ups on its first day and was soon scoring as many as 200,000 page views a day. A management and technical team was then formed to create the corporate platform needed to help NeoPets expand. They added more staff and moved its servers to Pixelgate, a Westlake Village, California-based web hosting and internet services company. “After being off-line for several days, we surpassed 600,000 page views within three days of getting back on-line,” said NeoPets Chairman and CEO Doug Dohring.

The company increased the number of Apache/Linux boxes from two to five, using single CPU P3-600s as image servers and dual P3-600s for web servers, each with 512MB to 1GB RAM. Continual load expansion eventually pushed NeoPet's MySQL database technology to the limit. By this time, NeoPets was surpassing up to ten million page views a day. Reorganization again became a necessity.

The company secured the services of Web Zone Inc. of Santa Clara, California and Broomfield, Colorado-based Level 3, a multinational Tier One provider with hosting facilities in Los Angeles. This provided enough bandwidth to deal comfortably with anticipated traffic volumes. NeoPets then added yet more staff and purchased about 50 Red Hat/Apache web and image servers, two more MySQL Servers and a Sun server to run an Oracle database. Once the Oracle conversion was completed, page views soared to over 40 million a day.

The current NeoPets architecture comprises a Red Hat 6.2 and Apache front end, with a Solaris and Oracle back end. At the same time, MySQL is still used for a wide range of database operations.

Despite the introduction of Oracle, NeoPets remains one of the larger users of Apache on the Web. Though Oracle had to be introduced to provide a heavy-duty database, NeoPets believes that open source ultimately offers better quality and greater product reliability and remains committed to further expanding the robustness and capacity of PHP, Apache and MySQL as an alternative to Oracle.

“We are looking for people who can modify these open-source applications and take them to a new plateau,” said CTO Bill McCaffrey. “If we involve the right people, we believe that we can take these applications to the point where they can be used for even the largest sites on the Web.”

In anticipation of another summertime boom in site usage, NeoPets is planning to add many more web developers and open-source programmers, as well as system administrators and IT support staff.

—Drew Robb

The *Linux Journal* Award for Applied Calimari Goes To—

Open source is a fine development model, but with the obvious exception of Eric Raymond it kind of sucks at PR.

Okay, let's qualify that. There are some fine companies that get mileage out of open source as a virtue, but as an editor I can tell you that there are too darn few pure open-source projects .org-type with a PR department (we suspect that number is zero), or with much PR instinct, by which I mean they bother editors like me with interesting information about what they're up to. Sure, we get flamed to a cinder when we neglect to mention the obvious, such as early last year when we wrongly reported that Borland's InterBase was about to become the first open-source database project, earning the outrage of some PostgreSQL folks (though surprisingly few, considering). But there isn't much outreach by the growing assortment of nuts-and-bolts open-source projects that simply make something handy that a lot of others can use.

Take proxy caching, which is very handy if you've got a lot of traffic to manage —but not much of a conversation starter except for those who (for professional or other reasons) obsess about it.

As it happens there are more than a few obsessives out there, and one of them (I forget who) told me that Squid (http://www.squid-cache.org/) is the cat's pajamas of open-source proxy servers. Well, it seems there are a pile of proprietary (presumably closed-source, certainly not free) proxy servers in the world. You can get them from Lucent, Novell, IBM, Cisco, Microsoft and the other usual suspects. Their prices run from zero to six figures. Squid is at the bottom of that range. As their FAQ puts it, “You can download Squid via FTP from the primary FTP site or one of the many worldwide mirror sites. Many sushi bars also have Squid.”

The product is competitive—literally. A group called IRCache holds frequent bake-offs (which they now call cache-offs) using the web Polygraph (http://www.polygraph.ircache.net/), a benchmarking tool developed by the National Science Foundation and a bunch of those same usual suspects. The results (also on the IRCache site) for each bake/cache-off run through many pages, many tables and many graphs. Squid leads in some places and lags in others, but it runs in the thick of every race.

Perhaps the most telling results come from this level-5 post from Matthew P. Barnson on Slashdot last year:

> I can personally say that the three I've had experience with, Novell's ICS caches (which comprised ten of the twenty entrants), Network Appliance's NetCache, and Squid (on Solaris, in our case) all rock. Squid 2.3-stable1 was a dream to compile, install, and configure.

When we contacted him directly, he added this about Squid: “As an outgrowth of the Harvest Project, this venerable, free-software proxy cache sets the benchmark by which all other caches are measured.... For the price, Squid kicks some serious butt!” He also has kind words for another open-source project:

> Apache web server was not specifically mentioned in the bake-off, but in my experience is extremely popular for caching services because the same server that can serve your web pages from your dorm room can also speed up your web surfing.

And he should know his stuff, he works at one of the most heavily visited sites on Earth (iMALL.com).

So let's raise a glass of saki to the Squid team and invite all the other open-source and free-software developers who envy this kind of coverage to let us know what they're up to.

Stop the Presses

What's happens when the Penguin-in-Chief himself issues an oh-by-the-way e-mail to the Kernel Mailing List that takes the form of a halfhearted (dare we say half-minded?) and clearly tongue-in-cheek press release that just happens to be entirely about the long-awaited version 2.4 of the Linux kernel?

A worldwide sigh of thanks, followed by traffic jams at the ftp servers.

It went like this:

> Date: Thu, 4 Jan 2001 16:01:22 -0800 (PST)<\n> From: Linus Torvalds torvalds@transmeta.comTo: Kernel Mailing List linux-kernel@vger.kernel.orgSubject: And oh, btw...
>
> In a move unanimously hailed by the trade press and industry analysts as being a sure sign of incipient brain damage, Linus Torvalds (also known as the "father of Linux" or, more commonly, as "mush-for-brains") decided that enough is enough, and that things don't get better from having the same people test it over and over again. In short, 2.4.0 is out there.
>
> Anxiously awaited for the last too many months, 2.4.0 brings to the table many improvements, none of which come to mind to the exhausted release manager right now. "It's better", was the only printable quote. Pressed for details, Linus bared his teeth and hissed at reporters, most of whom suddenly remembered that they'd rather cover "Home and Gardening" than the IT industry anyway.
>
> Anyway, have fun. And don't bother reporting any bugs for the next few days. I won't care anyway.
>
> —Linus

Context: The kernel had a bit of a Y2K problem of its own. In January 2000, Linus said the 2.4 kernel would be out in the summer. Then in November he said the kernel would be released in early December. Now here it is, pretty much exactly one year, um, later. And hey: so what?

The features? USB support, symmetric multiprocessing support, a rewritten networking layer, driver updates, other good stuff.

2.6 is next. Let the wait begin.

—Doc Searls

They Said It

Never attribute malice to what can be adequately explained as pure-unfiltered-idiocy.

—Joseph E. Arruda

Black holes are where God divided by zero.

—Steven Wright

Eagles may soar, but weasels don't get sucked into jet engines.

—Anonymous Psychopath on Slashdot

Any technology distinguishable from magic is insufficiently advanced.

—Don Marti (as far as we know)

Every revolution has been preceded by hard critical thinking, the diffusion of culture, and the spread of ideas among men who are at first unwilling to listen, men concerned with solving their private economic and political problems.

—Antonio Gramsci.

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

—Bertrand Russell

It is a myth that people resist change. People resist what other people make them do, not what they themselves choose to do. . . . That's why companies that innovate successfully year after year seek their people's ideas, let them initiate new projects and encourage more experiments.

—Rosabeth Moss Kanter

Long-range planning does not deal with future decisions, but with the future of present decisions.

—Peter F. Drucker.

You can't depend on your judgment when your imagination is out of focus.

—Mark Twain.

Discovery consists of seeing what everybody has seen and thinking what nobody has thought.

—Albert Szent-Gyorgyi.

The Internet never retreats.

—Vint Cerf

Linux (le-nuks, lin-uks) noun. A version of the UNIX System V Release 3.0 kernel developed for PCs with 80386 and higher microprocessors. Developed by Linus Torvalds of Sweden (for whom it is named).

—from Microsoft Bookshelf (spotted by Wojciech Tatina)

The only piece of software I've never cursed is emacs. It changes modes effortlessly. When I'm editing a Perl script it adds the tags and checks the parens. When I edit a letter it gives me all the carriage returns in the right place. It's one piece of software, but it understands file extensions. emacs knows what I'm up to. It's okay with what I do and it tries to help. I often find Word trying to add bullet points or numbers where I don't want them. emacs never does that to me. Of course emacs and I grew up in the same environment, so maybe that makes sense.

—Clay Shirky

...being a Linux user is sort of like living in a house inhabited by a large family of carpenters and architects. Every morning when you wake up, the house is a little different. Maybe there is a new turret, or some walls have moved. Or perhaps someone has temporarily removed the floor under your bed.

—John R. Levine and Margaret Levine Young

To try to do something that is inherently impossible is always a corrupting enterprise.

—Michael Oakshott

Bored people are the best consumers.

—John Taylor Gatto

You people just don't get it, do you? All Linux applications run on Solaris, which is our implementation of Linux.

—Scott McNealy

Best of Technical Support

Various

Issue #83, March 2001

Our experts answer your technical questions.

Configuring the Mouse Wheel, Three Ways

I'm wondering if it is possible to config the scrollable mouse to work in Linux like it works in Windows. The mouse I'm using is the Microsoft optical intellimouse. —Ray, wrathstocks@ragingbull.com

Sure! Using the **imwheel** command you will be able to do it. **imwheel -?** will tell you the correct syntax. For more information, see jcatki.dhs.org/imwheel.

—Paulo Wollny, paulo@wollny.com.br

Yes. You will find all the information you need at http://www-sop.inria.fr/koala/colas/mouse-wheel-scroll/.

—Marc Merlin, marc_bts@valinux.com

You can set this up in your /etc/X11/XF86Config file. Edit the Pointer section so that protocol is “imps/2”. Below that add these lines:

```
Option   "Emulate3Buttons"   "off"
Option   "ZAxisMapping"   "4 5"
```

Then restart X and it should work.

—Paul Christensen, pchristensen@penguincomputing.com

termcap for Connecting to SCO

We have a FoxPro 2.6 for UNIX running in SCO Open Server Release 5. The users access this application from Windows using Tiny Term. We want to move these users to Linux, so far we tried Telnet, defining TERM as scoansi and several other terminals and rlogin with xterm, but we cannot get the right key mappings.

—Jorge C. Oneto, jconeto@jco.wamani.apc.org

According to SCO, many Linux distributions contain incorrect scoansi termcap entries. I believe SCO would be willing to send you a new termcap entry that will fix the problems you are seeing. (Without more details than “cannot get the right key mappings” it's hard to say what exactly is wrong.)

—David Brown, david@caldera.com

I did a migration of a FoxPro application from SCO UNIX to Linux and we had to manually modify a terminfo entry to get a 100% scoansi compatible terminal emulation. After modifying the most adequate (probably vt100 or ansi) terminfo file you will need to compile it using the tic terminfo compiler. Do a **man tic** to get more information about this command. I would suggest that you look at an actual scoansi emulation definition file on a real SCO system and take a close look at the escape sequences definition so you can port them to Linux (beware of copyright issues if you plan on just copying this file from the SCO system to the Linux system). Another good approach that helped me with the keyboard part is the **xmodmap** command that lets you define sequences for each key to be sent from the keyboard to the host on an X terminal session.

—Felipe E. Barousse Boué, fbarousse@piensa.com

Bad Password, Bad, Bad!

What controls the composition of a password? How does the system enforce rules on length and mandatory use of non-alphanumeric characters.

—Westley L. Hespeth, hespethw@all-speth.com

For most Linux installations, it is a package called **cracklib.** PAM (plugable authentication module) uses cracklib to check the validity of a password by using cracklib (usually a special **pam_cracklib.so** or the like).

—David Brown, david@caldera.com

Unauthorized Hosts Trying to FTP

I am connected to the Internet with a static address. I have been seeing unauthorized FTP access to my system. Even though I have hosts.deny set to ALL:ALL.

The other thing I see is constant probes to tcpd port 1994 from **user unknown**. These connects look like this:

```
Nov 21 11:58:10 ns1 tcpd[1994]: warning:


Nov 21 11:58:10 ns1 tcpd[1994]: connect from unknown
Nov 21 11:58:10 ns1 tcpd[1994]: warning:


Nov 21 11:58:10 ns1 tcpd[1994]: connect from unknown
```

Any advice on plugging these holes?

—Dave Price, davep@support-one.com

It looks like tcpd is simply logging rejected connection attempts. In other words, your hosts.deny is doing its job. A much better and more flexible way to secure a machine is with ipchains. If you're interested check out the ipchains HOWTO at linuxdoc.org.

—Paul Christensen, pchristensen@penguincomputing.com

Using Second CD-ROM Drive in KDE

I installed the Caldera's distribution desktop 2.4 in my computer as a second OS. It's working fine.

I installed a second CD-ROM reader (ATAPI). The new reader and the ancient appears in Windows 98. But in Linux I found only the first CD-ROM. What do I have to do to declare my second CD-ROM? Is it possible to do it with KDE?

—Abderrahmane Meskine, ameskine@finances.gov.ma

You'll need to know where the CD-ROM is attached. After booting, log in as root and run:

```
dmesg | egrep 'hda|hdb|hdc|hdd'
```

(assuming it is an IDE device). Then look for your CD-ROM drive(s) in the output. Once you know the device to which your CD-ROM is attached, you can create a new device icon on the KDE desktop that points to your CD-ROM. Rightclick on your current CD-ROM icon and see what the settings are there. Then make a new device icon that is identical to the first in all settings except the actual device string. Change the device string to match what you found from the dmesg... command above (which will be something like

—David Brown, david@caldera.com

Cron Job without Mail from cron

I have a crontab running every ten minutes that does a two-packet ping to keep my network connection alive (why? it's a long story).

The problem I'm faced with is the number of e-mail notifications telling me that the command ran successfully. Is there a way to turn that feature off so that I'm not notified? This generates 144 e-mails/day.

—Scott A. Morrison Markham, scottm@ca.ibm.com

Sure, use this in your **/etc/crontab** instead.

```
ping -c 1 target &>/dev/null
```

—Marc Merlin, marc_bts@valinux.com

Telnet Doesn't Work? Good.

Hi, I am using Mandrake 7.1. After the installation I am unable to Telnet or FTP to my local machine.

```
telnet localhost<
  telnet:Error message—unable to connect to
         remote host: connection refused
```

—Pierre, pierre_poo@hotmail.com

Teaching people how to set up Telnet or ftp servers is irresponsible, so we won't do it. Shred your dusty old no-sense-of-security Internet books that explain these two insecure protocols (don't give them to a library; a kid might see them) and install **ssh**. Most distributions have easy-to-install ssh packages now. If you need to log in to your Linux box from non-Linux clients, check Rick Moen's canonical list of ssh software at: linuxmafia.com/pub/linux/security/ssh-clients.

—Don Marti, info@linuxjournal.com

Need SMP kernel

Do you know if there is a kernel available that supports multiple CPUs? I am using a dual PII 300 but only one of the CPUs is recognized.

—Crist Besore, cbesore@kuhncom.net

You should configure the kernel to enable SMP support:

```
cd /usr/src/linux
make menuconfig
```

Select “Processor type and feature” then select “Symmetric multi-processor support”.

—Pierre Ficheux, pficheux@wanadoo.fr

Configuring a Sound Card

I have read all of the README, FAQ and HOW_TO references on configuring a sound card but still am confused as to whether this is a necessity that requires rebuilding the kernel (which I have never done and am a bit nervous about). I detect a SB 3.01 card when I boot but then get an error message about the card being several years old or needing to be reconfigured. Would it be easier for me to go out and purchase a Linux-friendly card?

—Greg McNichol, gc@mcnichol.net

That depends on your card. If it's an ISA-based sound card, it is probably not going to work well unless it's one of the more popular cards, such as a SoundBlaster (NOT a SoundBlaster-compatible). But if it's PCI, you shouldn't be getting that message. In general, most PCI devices should be expected to work with Linux at this point. There are rare exceptions, but they are usually in the area of SCSI and network cards. One thing you should make sure is that there isn't another driver made specifically for your card, because the first driver that THINKS it knows what it's doing is allowed to control a device. This may be bad if your card isn't actually a SoundBlaster or clone, but the driver thinks it is. You can do this by rebuilding your kernel and looking through the sound card options, seeing if any matches it.

—Chad R. Robinson, crobinson@rfgonline.com

Linux on IBM A20p?

I'm a laptop user with an IBM A20p. I can't seem to find any documentation on the Web about installing Linux on my notebook. I hope you can provide me with some aid. Most importantly, I would like to know about the problems I will be facing, like hardware, config, etc. Any help in any form will be greatly appreciated.

—Mikael Koh, dragulako@yahoo.com

There is an install page for your laptop here: www.zhlive.ch/zhl_contents_linux.html

—Marc Merlin, marc_bts@valinux.com

This laptop has been certified by LinuxCare to run well with SuSE 6.4. You can find more information at: www.linuxcare.com/labs/certs/pada20p-suse64-sys.epl and www.zhlive.ch/zhl_contents_linux.html.

—Paul Christensen, pchristensen@penguincomputing.com

PPP Yes, Mail No

When adding a new user, e-mail is automatically granted. What I would like to do is specify a group that when a new user is added, e-mail is not given. An alternative would be removing a user's ability to send or receive e-mail—which I also am unable to find information on. Truly, I'm looking for a PPP.only account.

—Len Elyea, lelyea@ncmc.cc.mi.us

That's a bit hard to do right actually. For incoming e-mail you can alias the accounts to a dummy nonexistent account called nomail. For outgoing, I can't find any good solutions. If you were using **exim** instead of **sendmail**, you could have a list of PPP users in a file, and do header rewriting so that the From: and To: headers are rewritten to a nonexistent user, and then enable sender verification (X rewriting rule) to prevent nonexistent users to send mail. That said, this doesn't prevent someone from sending mail by connecting to a remote open relay.

—Marc Merlin, marc_bts@valinux.com

Slow Terminals

When I use any of my terminal emulation programs (GNOME, KDE) the response time is super slow, so slow it makes it very hard to work on the machine. Everything was working fine until yesterday. Could it be something with the display setup? I just can't seem to figure it out. All of the GUI programs work normalllyy. Help!

—Todd Ann Carter, tcarter@healthnetworkamerica.com

If the slow response time issue is at the login time, you may have a host name resolution problem. Try setting the /etc/hosts file with an entry with the IP address and fully qualified domain name of your machine and the line **order hosts, bind** in the /etc/host.conf file or, correctly setup DNS resolution. On the other hand, have you tried logging it at the console with NO graphics environment? If you can get into the system, try issuing a “ps ax” command to find out which process is taking up so much resources and take corrective actions from there.

—Felipe E. Barousse Boué, fbarousse@piensa.com

Installing Netscape 6 on Red Hat

I'm very new to Linux. I need to install Netscape 6 browser on my Red Had 7.0 box. I did unzip file, downloaded from Netscape and put them under /usr/ My_Downloads/netscape 6. What should I do next? I tried double clicking on Netscape-installer, but this does not work. What command should I use to install Netscape 6? Please help.

—Alex, aqk13@hotmail.com

After you've downloaded and unpacked the Netscape 6 tar file, cd into the netscape-installer directory and type

```
   ./netscape-installer
```

An installation wizard should appear giving you options of what to install and where the installation is going to be placed on your system. Make sure to do this as root.

—Paul Christensen, pchristensen@penguincomputing.com

New Products

Heather Mead

Issue #83, March 2001

Wing IDE for Python, NetMAX VPN Server Suite, iConnect Suite and more.

Wing IDE for Python

Wing IDE is an integrated development environment for the Python programming language. Software developers are provided with an integrated product manager, graphical debugger, source-code browser and a source-code editor. The IDE works with Makefiles and other external build configuration tools. It also provides support for debugging programs written using the Tkinter, PyGtk and PyQt toolkits.

Contact: Archaeopteryx Software, Inc., PO Box 1937, Brookline, Massachusetts 02446-0016, 617-232-0059, info@archaeopteryx.com, http://archaeopteryx.com/.

NetMAX VPN Server Suite

Netmax VPN Server Suite

Geared toward small-and medium-sized businesses, the server suite allows users to connect their networks securely to remote sites, business partners and telecommuters using the Internet. The suite includes IPSec protocol for encryption and traffic routing between networks, the triple data encryption standard, automatic key management, 128-bit SSL, prepacket authentication using SHA-1 or MD5 and a full version of the MetMAX Firewall ProSuite.

Contact: NetMAX, Cybernet Systems Corporation, 726 Airport Boulevard, Ann Arbor, Michigan 48108, 800-292-3763 (toll-free), info@netmax.com, http://www.netmax.com/.

iConnect Suite

iConnect is a web-based suite of tools that offers solutions for businesses of any size to web-enable applications. Applications installed on a remote server can be accessed over the Internet using a standard Java-enabled web browser. Services in the suite include HTTP-based applications and data management services, file transfer, distribution and synchronization; client application and management; and Web-based shell environments, job management and control. Evaluation copies are available for download.

Contact: SoftKnot Corporation, 46500 Fremont Boulevard, Suite #716, Fremont, California 94538, 510-226-2768, mmovahed@softknot.com, http://www.softknot.com/.

Linux2order Web Site

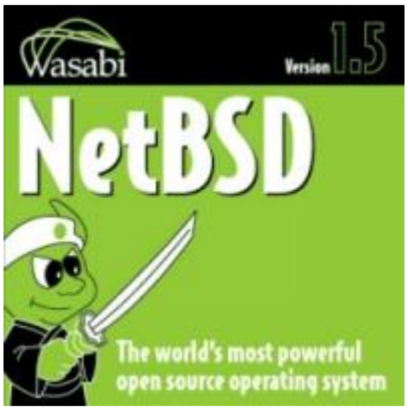
Linux2order™
The Source for Linux Applications

Linux2order Web Site

The Linux2order database contains over 9,000 open-source applications and utilities on the web site for download or compilation. Titles are updated daily to incorporate the most recent builds and releases. Descriptions and reviews of programs are also available. Users can either download selected files directly or request the creation of a custom CD-ROM containing their chosen files.

Contact: Linux2order, 5252 North Edgewood Drive, Provo, Utah 84604, 801-222-9414, http://www.linux2order.com/.

NetBSD 1.5 CD-ROM



NetBSD 1.5 CD-ROM

NetBSD 1.5 on CD-ROM is a fully functional UNIX-like OS. Features include a complete set of user utilities, compilers for several languages, the X Window System, firewall software, support for wireless and plug-and-play peripherals and full source code. Benefits of NetBSD include software portability, coherency between platforms, highly developed networking and high-level emulation. Wasabi's distribution comes with on-disk documentation and an installation and support guide.

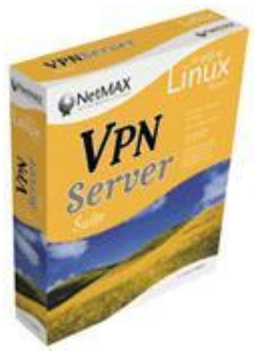
Contact: Wasabi Systems, Inc., 104 West 14th Street, 4th Floor, New York, New York 10011, 757-248-9601, info@wasabisystems.com, http://www.wasabisystems.com/.

WARP Performance Suite

The WARP Performance Suite of Web infrastructure software was developed for optimum performance of Internet applications. Improving performance at the origin of web content is the goal. The initial components of the suite are the Intelligent Content Distributor, Load Balancer and Global Load Balancer; the Dynamic Content Director, Cache Master and Secure will be added in the first quarter of 2001. Availability is on a monthly subscription basis.

Contact: WARP Solutions, Inc., 627 Greenwich Street, New York, New York 10014, 877-688-WARP (toll-free), info@warpsolutions.com, http:/www.warpsolutions.com/.

INTERNETpro Small Enterprise Server 2012



INTERNETpro Small Enterprise Server 2012

INTERNETpro Small Enterprise Server 2012 was created for virtual private networking (VPN) that coordinates with dynamic IP addressing. For use with e-commerce and telecommunications, the 2012 includes VPN with advanced encryption, firewall, web server, e-mail server, built-in routing, WAN connectivity with multiple options, web caching, proxy server, URL blocking and dynamic content screening. The 2012 has an 8GB hard drive and 450MHz CPU to support up to 150 concurrent users.

Contact: Internet Appliance, Inc., 40515 Encyclopedia Circle, Fremont, California 94538, 510-413-1068, sales@internetappliance.com, http://www.internetappliance.com/.

ProcureMind 1.6

ProcureMind 1.6, from Mindflow Technologies, is software for formulating and tracking purchasing strategies designed to take the place of spreadsheets. Capabilities for entering, viewing and manipulating data in a variety of formats are provided in one product. Improvements to the component programs, ProcureStrat, ProcureSave and Smartsourcing Desktop in version 1.6, include multiline optimization for bidding analysis. A sampling of analysis capabilities include multi-item bundling efficiencies, supplier relationship management and increased sourcing productivity.

Contact: Mindflow Technologies, Inc., 6504 International Parkway, Suite 2400, Plano, Texas 75093, 972-930-9988, info@mindflow.com, http://www.mindflow.com/.

MSC.Linux

MSC.Linux is a new distribution offered by MSC.Software Corporation that combines PCs into clusters, enabling them to address engineering problems usually handled by supercomputers. Features of the Web-based distribution include substantial memory, high-performance data transfer and cluster tools. MSC.Linux is aimed at the product-design market to enable increased time-to-market efficiency. The beta release is available for download from the web site (http://www.mscsoftware.com/).

Contact: MSC.Software Corporation, 815 Colorado Boulevard, Los Angeles, California 90041, 323-258-9111, http://www.mscsoftware.com/.