

Advanced search

Linux Journal Issue #85/May 2001



Features

Focus: Training and Certification by Richard Vernon
Training Digital Divide Warriors by Ana Maria Harkins

The skinny on training techno missionaries.

An Accurate Assessment? by Richard Morgan

A review of the LPIC process.

Why Be Certified? by Tobin Maginnis

The value of certification.

Indepth

Using Python to Query MySQL over the Net by Mihai Bisca

Use Python to upgrade your site's search.

MOSIX: A Cluster Load-Balancing Solution for Linux by Ibrahim F. Haddad and Evangeline Paquin

A clustered Linux telecom-grade internet server.

Apache Toolbox by Ralph Krause

The Apache Toolbox simplifies installation and configuration of the Apache web server.

Boot with GRUB by Wayne Marshall

Run multiple OSES with free GRUB.

Toolbox

Take Command An Introduction to DNS and DNS Tools by Neil Anuskiewicz

Kernel Korner [Linux Teleconferencing: Improving the Wireless Network](#) by Izzet Agoren

At the Forge [JavaServer Pages](#) by Reuven M. Lerner

Cooking with Linux [Fly Me to the Wine Cellar](#) by Marcel Gagné

Paranoid Penguin [Checking Your Work with Scanners, Part I \(of II\): nmap](#) by Mick Bauer

Columns

[Focus on Software](#) by David Bandel

Linux for Suits [Adjusting to Life in the Bazaar](#) by Doc Searls

Focus on Embedded Systems [The embedded side of LinuxWorld](#) by Rick Lehrbaum

Games Penguins Play [SimCity 3000 Unlimited for Linux](#) by Neil Doane

.org Watch [A call to action](#) by Leslie Proctor

Reviews

[The PC Weasel 2000](#) by Jon Valesh

[Core Python Programming](#) by Michael Baxter

[Professional Linux Programming](#) by Stephanie Black

Departments

[Letters](#)

[upFRONT](#)

[Best of Technical Support](#)

[New Products](#)

[Archive Index](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Focus: Training and Certification

Richard Vernon

Issue #85, May 2001

This month's feature articles show that even with the additions that accompany hitting the big time, the Linux community has maintained a balance between the scramble for cash and keeping the philanthropic element as far from moribundity as possible.

The market for certification programs for Linux professionals, such as the Linux Professional Institute, Sair and Red Hat's RHCE, is evidence that Linux has gained a sure foothold in the business and computer-professional world.

However, many Linux enthusiasts feel that as the use of the open-source OS moves into the realm of mainstream business and attracts the development and marketing dollars of large corporations it is losing some of the qualities that make it attractive, that the addition of both big(ger) business and money to the Linux community adulterates what began as a hobbyist's free, and dare I say ethical, operating system.

The mix of this month's focus articles on training and certification shows that even with the additions that accompany hitting the big time, the Linux community has maintained a balance between the scramble for "filthy lucre" and keeping the philanthropic element as far from moribundity as possible.

Linux is about accessibility—accessibility to software, code and technical advice and support from the community of users. Unfortunately, most of this support is available only on the Internet. Geekcorps is a nonprofit that takes Linux support, or rather education, to the doors of those who need it most. Ana Maria Harkins, in her article on page 80 describes it as a "Peace Corps for techies". Their volunteers are presently working hard at making Linux accessible to those in Ghana, and Ana Maria describes what it takes to be, and to train, a Geekcorps volunteer.

An official certification of Linux competence can speed the adoption of the operating system by dispelling the perception that there is a lack of people who comprehend it. "Linux certified" on résumés, clues employers in to the fact that there are people who know it, and at the same time it encourages job seekers to learn it to sharpen the edge on their résumés. As Don Marti says, "Certification programs are a powerful self-reinforcing consensus reality for promoting an OS, even if they don't teach anything. Any learning you get in the process is pure gravy."

Tobin Maginnis, president of Sair Linux and GNU Certification, gives some additional good reasons for Linux system administrators to have their knowledge certified (page 84). As he points out, the growing demand for Linux certification is a validation of the work of Linux developers and evangelists to move the operating system into more widespread usage. Maginnis also explains Sair's methods and vendor-neutral philosophy.

The LPI (Linux Professional Institute) is a dot org that demonstrates the active spirit of the Linux community through its reliance on community input. Richard Morgan, after completing the LPI certification first-level exam, shares his thoughts and comments as to why the LPI process has room for improvement but is still one of the best ways to improve one's résumé (page 86).

—Richard Vernon, Editor in Chief

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Training Digital Divide Warriors

Ana Maria Harkins

Issue #85, May 2001

The nonprofit Geekcorps teaches businesses in emerging countries to use technology while providing service volunteers opportunities to share their skills in unfamiliar circumstances.

As we approach Geekcorps' one year anniversary of incorporation, it seems impossible that two waves of Geekcorps volunteers have already been sent to Ghana. Conceived as a Peace Corps for techies, this nonprofit organization has flung itself head over heels into the mire of conquering the international digital divide. Volunteers from the high-tech industry in the US and Europe spend four months helping businesses in developing countries—Ghana at the moment—to utilize ICT (information/communication technology) to improve their core operations.

So who wants to do this, and what kind of person does Geekcorps look for? When I interviewed Peter Beardsley last October for a volunteer position in Ghana, I thought “this guy would be great.” Peter grew up in Connecticut and went to college in New Hampshire where he studied programming. After discovering Linux through a friend, he worked on web scripting and applications for his college as an independent study project. Aside from his technical skills, what made Peter a great potential volunteer was his demonstrated interest in helping people out (fixing up old machines and turning them into servers and NATs for churches and community groups through his local Linux users' group) and his belief that technology can truly bring about social and economic change. Peter has both a love of technology and a willingness to make it accessible to everyone.



Geeks pose outside Geekcorps' headquarters in North Adams, Massachusetts before traveling to Ghana.

How does one train a guy like Peter to go in to an unfamiliar country and culture and transfer his skills (database development) to people whose knowledge of IT is often spotty? From the inception of Geekcorps, training has been an important component of the program. We had seen many projects where people were sent out unprepared and uninformed. Unlike the Peace Corps, our volunteers are only at their assignments for a relatively brief time. We do not have the luxury of providing weeks of cultural immersion and seminars on the politics and history of their host country. But because their stay is so short, we can't have them spend the bulk of their time in a state of disorientation. Our first two training programs have taken place at our offices in the US and have lasted about a week and a half. An additional in-country orientation focuses more on the practicalities of living and working and lasts about five days.

The Geekcorps training program is still (and will probably always be) a work in progress. We start from the premise that volunteers do not need subject-matter training (i.e., we don't go over programming or web design topics). What they do need is a hefty dose of cross-cultural training, discussions centering on technology and development, and training of teaching exercises and expectation management.

We have discovered that this last item is key. The training of our second wave of volunteers centered on our expectations of them, their expectations of themselves and their counterpart businesses' expectations. In most instances, volunteers have unrealistic expectations: "my counterpart company will be listed on the NASDAQ when I finish", "having people approach me on a daily basis asking for money will not bother me", or "I have a stomach of steel".

Training skills have also turned out to be critical. At Geekcorps headquarters in North Adams, Massachusetts volunteers review adult learning cycles and learning styles and follow up with a hands-on training that involves teaching basic computer skills. Hands-on training gives volunteers the opportunity to re-

examine their approach to training and learning, especially at the basic skill level. While their counterparts in Ghana are beyond the basics, their understanding of technology is often piecemeal or weighted in one specific topic. Volunteers need to transfer their skills thoroughly enough so as to be sustainable, and dynamically enough to keep people excited about the practical applications that will improve their businesses.

After they arrive in Ghana, training is taken up by our country director, Stophe Landis. Volunteers are no longer in their comfort zone and begin to understand what they have gotten into. This portion of the training focuses on concrete work and culture issues. An example is the scavenger hunt volunteers play on their second or third day in-country. Volunteers are instructed to find a place in Accra (the capital of Ghana) such as a landmark, restaurant or company, with incomplete information and little guidance.

To a great extent, Geekcorps volunteers are self-selected risk takers ("sure, I'll entrust you with my life!"), jacks-of-all-trades ("sure, I can make this laser pointer into a scanner that runs on biogas!") and theory-oriented. Thus the training has to address the needs of this particular subset of traits. I realized this one afternoon when I was facilitating a culture exercise titled "Universal-Cultural-Personal". The exercise is used to clarify what behaviors can be categorized under the above headings. I found myself entering into an argument that threatened to escalate into a screaming match because several volunteers claimed that eating on a regular basis can be construed as a cultural, not a universal trait. I realized that while this obfuscation of the exercise frustrated me, the volunteers really liked turning the construct around and looking at it from all angles. Why? Just because it seemed interesting from a theoretical standpoint.

What hopefully emerges out of the Geekcorps training is a volunteer who is more self-aware, more tolerant of ambiguity in every realm of his or her own life, better able to cope with things like no running water, able to live without T1 connectivity, able to help businesses achieve a solution that is appropriate to that environment and not afraid to engage with the people around them, no matter how unfamiliar.

You can read more about the training program, the work projects and sundry extracurricular activities from the volunteer's point of view at their web site (<http://www.geekhalla.org/>). You can also visit Geekcorps' web site (<http://www.geekcorps.org/>).



Ana Maria Harkins is the director of training and recruitment at Geekcorps. Prior to her current exalted status, she was a humanitarian relief worker with a refugee agency, a researcher for the Smithsonian Institution and a lackey at several other nonprofit organizations.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

An Accurate Assessment?

Richard Morgan

Issue #85, May 2001

A review of the LPIC process.

As the Linux community has grown from the world of the hobbyist to include professional administrators, the infrastructure of the community and marketplace has evolved to support this rapid growth. We now have numerous Linux periodicals, reference books, web sites and significant corporate involvement. In recent months, the idea of a certification process to quantify the skills of system administrators has also become quite popular in the Linux community. Numerous information technology companies such as Sun, Cisco, Novell and Microsoft have had popular certification programs for their own particular product lines for some time.

Several current certification programs try to gauge the skill level of Linux system administrators via testing. The more well-known programs include Red Hat's RHCE, Sair's GNU/Linux Certification and Linux Professional Institute's LPIC.

Recently I took the first-level Linux Professional Institute Certification exam (101). The following is a brief overview of the program and also a review of my experience with the exam and preparation materials.

My background is varied, but I have been working with UNIX systems in some limited form for about seven years, first doing data conversions and later working with database applications. I have been using Linux for about three and a half years. Primarily a Red Hat and Mandrake user, I have lately begun to fiddle with the Debian distribution in my spare time. Once an ardent user of RPM (Red Hat's package manager), I now see the beauty of Debian's apt-get commands. For more important programs, I tend to compile from source rather than using packages. But I don't compile and install every single released kernel; in fact, before the stable 2.4 kernel arrived, I hadn't compiled a kernel in months. For me, Linux is about the freedom and flexibility I find in using and

developing Linux applications daily, not loading the latest development kernel to look for bugs (though many people do this, and Linux is better for their efforts).

In my current job, I do a mix of programming (Perl for CGI, DBI and shell scripting), system administration and implementation. As the webmaster of the northern Virginia Linux Users' Group and Tux.Org, I get to dig into topics such as Apache web server configuration and DNS. At home, I once had a Linux box with 380 days of uptime and currently use Linux for all of my day-to-day work. This includes reading e-mail, writing code, compiling programs—pretty much everything. I have a small network there with a gateway machine and six machines behind it. In summary, I am comfortable with most basic administration topics.

My motivation in taking the certification test was two-fold. I hoped to quantify my Linux knowledge, which was gained with little formal training. Plus, the certification would surely make a nice addition to my résumé.

I chose the Linux Professional Institute Certification program over the other options because of its apparent vendor and distribution neutrality. The Linux Professional Institute has support and input from a wide array of sources. These include Caldera, IBM, Linuxcare, Silicon Graphics, SuSE, Hewlett-Packard and VA Linux Systems, among others. The LPI advisory council of over fifty members reads like a roll call of the well-known members of the community.

The LPI certification process consists of three levels, each requiring successful completion of two tests. Each level has a numeric designation for its tests, 101, 102, 201, 202 and so on.

LPI outlines a body of knowledge with specific skills at each certification level. For example, according to the LPI web site, to be successful with a Level 1 exam, you should be able to complete the following tasks: work from the UNIX command line; perform easy maintenance tasks such as helping users, adding users to a larger system, backing up, restoring, shutting down and rebooting; install and configure a work station (including X) and connect it to the LAN or a standalone PC via modem to the Internet.

As such, the Level 1 detailed objectives list addresses the technology behind these tasks and the mechanics of performing them. For example, part of the detailed objective for setting up the X Window System reads:

Obj 1: Install and Configure XFree86—Verify that the video card and monitor are supported by an X server, install the correct X server, configure the X server, install an X font server, install required fonts for X (may require a manual edit of `/etc/X11/XF86Config` in the

Files section), customize and tune X for video card and monitor.

Commands: XF86Setup, xf86config.

Files: /etc/X11/XF86Config, .xresources.

The full set of detailed objectives appears quite comprehensive and representative of the Linux tasks a system administrator might face on the job. The structure of the material on the LPIC web site was logical and straightforward. It provided a good system for easily reviewing a small chunk of related materials, rather than requiring great effort to pick out the relevant material.

At the time I began this process, there were two LPIC preparation books available. I purchased both, mostly to get a feel for the format of the tests and to use as a concise list of topics to be reviewed. These books were *General Linux 1 Exam Prep*, published by the Coriolis Group and *LPIC Prep Kit 101*, published by Que.

The Coriolis book (\$49.99 US) had good content for reviewing administration topics, and I even learned some neat things by reading it. The book provides a review of the topics in both the 101 and 102 exams, so I only had to read half of it for this first exam. Each chapter reviews its topics thoroughly and provides a practice test at the end. However, the practice tests are full of errors, typographical and otherwise. For example, in a question that has possible answers of A, B, C or D, the answer key states that the answer is 10. I got the feeling this book was really rushed to market. The accompanying CD-ROM of practice tests has a clunky and amateurish feel. I checked the Coriolis web site for an errata page for this book but did not find one.

The Que book (\$39.99 US) has clear, concise content, but the CD-ROM of practice tests is a technical abomination. For example, while taking a practice test, if you need to review or change a previous answer and then move forward to continue the test, the program falls into a nasty little loop. The same question is presented to you repeatedly, while the test program still expects the answers to the questions that should be there. The only solution is to kill the test program and start over. It was frustrating, to say the least.

Working around these technical hurdles, I did the practice tests twice, a few weeks apart, and noted my errors. I restudied the areas where my knowledge was weakest, read applicable man pages and tried out unfamiliar commands and tools.

Besides studying for the exam itself, I had to make an appointment at a local testing center and pay the required fee. Each of the six LPIC exams costs \$100 US. The LPIC exams are administered through VUE (Virtual University Enterprises), which also administers numerous other certifications exams.

To register, I used the VUE web site to set up a VUE login and password. The web site has a nice interface for choosing a nearby testing center and making an appointment for the exam. I paid the \$100 fee by credit card and received an exam appointment confirmation and payment receipt a few hours later via e-mail—all very simple and straightforward.

Having registered and paid the fee, I arrived on the assigned date to take my exam. I was given 90 minutes to complete a computer-based selection of 60 questions. The format of the questions included a mix of multiple-choice, “choose all answers that apply” and short answer fill-in-the-blank.

The 60 questions, apparently chosen at random, did not allow me to be tested evenly (or at all in some areas) on the many facets of administering a Linux system. I do not remember seeing a single question on how to back up a system. A certain number of questions from every section in the LPI “detailed objectives” would have been more accurate in testing my overall knowledge, even if it made the exam longer.

I felt the exam questions placed too much emphasis on knowing arcane and often little-used options for many system commands, though this may be the result of my particular type of experience. In my exam, there were numerous questions on text filters and the use of sed and regular expressions. In all honesty, I have yet to create a regular expression in my work without a couple of passes to get it exactly right.

The testing software (Windows-based) provided a comment mechanism, so I commented on many of the questions heavily, critiquing confusing wording and undefined scope. In general, I found the test questions confusing and, in many cases, downright misleading—something I cannot completely attribute to a lack of knowledge on my part. I hope others commented on the questions and that the comments made their way back to LPI for consideration.

Finally, I did not receive a grade when I completed the test. The attendant at the testing center handed me a page that told me the exam was in beta testing, and results would be mailed to me in 6 to 12 weeks. LPIC eventually scored the exam and mailed me the score four months after the test date. I did pass the exam, with a score of 540 on a scale of 200-800.

Conclusion

My knowledge is not where it could be, and I continue to learn every day. But, the exams still need considerable improvement to be considered a fair assessment of an administrator's skills. I believe the work is actively underway at LPI to improve the certification exams. Even with the minor difficulties, this certification program is still my choice over the others, mostly due to the wide support and vendor neutrality of the program.

Computerized testing doesn't allow one the luxury of fiddling with the options to obtain an exact result (like regular expressions) and thus was not a good mechanism for testing some areas of system administration, where bizarre and varied problems are encountered daily. Other exams may have a more practical approach in challenging you to fix a broken system within an allotted time period.

In summary, taking the LPIC exams can be a good personal assessment of system administrator knowledge even though quantifying that via a computerized exam may not be entirely perfect. Plus, I feel that certifications, along with strong and relevant experience cannot hurt your chances in our competitive world. I recently registered for the 102 exam to be taken soon.

Resources



Richard Morgan (rmorgan@tux.org) lives on a mountain and works in northern Virginia as a systems engineer. When not at work or spending time with his wife and daughter, he hides in the basement with his computers doing fiendish things. He contributes his free time to the northern Virginia Linux Users' Group and Tux.org as their webmaster, along with tracing his ancestors back some 500 years. You can visit him on the Web at <http://www.tux.org/~rmorgan/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Why Be Certified?

Tobin Maginnis

Issue #85, May 2001

Tobin Maginnis, Sair president, provides some arguments in favor of Linux certification.

William Shakespeare penned the famous line in *Hamlet*, "To be or not to be. That is the question." If you are a Linux disciple, chances are you ask that same question today with an open-source twist: "To be or not to be certified. That is the question." The fact is, the Linux community is somewhat divided on the answer to this question.

One camp, often referred to as the "live free or die" followers, believes there is no reason to be certified. Their assumption is that because Linux—unlike such competitors as Microsoft and Novell—is open-source software that can be downloaded free off the Internet, knowledge and practical skill development come with hands-on, job-related experience and will not necessarily be bolstered by certification.

An opposing view, however, is that certification means validation. If a third party, or independent source, tests your skills and finds you possess the knowledge to run a complex operating system—one that is predicted to grow faster over the next four years than all other operating systems combined, it can benefit both the IT professional and the company that hires him or her.

The problem is not unlike owning a \$100,000 Porsche and trusting it to a certified service representative over a gas station mechanic with no concrete credentials.

Certification means money, both to the company and the certified employee. The benefits to the company are numerous. Research by International Data Corporation (IDC) shows IT managers trained and certified in Microsoft, Novell and Cisco programming have a positive impact in the workplace. The same will hold true with certified Linux professionals.

The certified shop is apt to be more efficient and productive. A certified IT employee can usually configure a desktop or server in a more timely manner than one who is not certified. The properly trained Linux programmer will be able to keep the server up and running, while some noncertified employees may depend on trial and error.

A certified shop can traditionally handle twice as much equipment as a noncertified shop, and a certified shop of employees can usually create a much more elaborate and complex network configuration.

In general, the cost of training pays for itself in just nine months. Additionally, certification is frequently used by IT and human resource managers to weed through stacks of résumés.

For the certified employee, benefits include higher salary, more job satisfaction, job security and greater knowledge that leads to more independence. More often than not, *numero uno* in the employee's book is better pay. Studies have shown certification in a desired skill such as Linux will lead to salary increases of \$5-\$10 per hour over noncertified peers. That translates to a \$10-\$20,000 increase per year in salary. Some find that certification also brings a sense of accomplishment and greater job satisfaction. So why Linux instead of those proprietary operating systems?

As already mentioned, Linux has emerged as the number two operating system worldwide and is predicted to grow faster than all other operating systems combined. A recent poll of more than 2,000 IT managers by Survey.com, an on-line research service, indicated a planned 500% increase in applications development for Linux over the next two years. International Data Corporation reports that while Linux is not a major market at the moment (estimated at \$56 million for 2001), Linux support revenue will increase to \$285 million in 2004, a compound annual growth rate of 86.9%.

Louis Gerstner, IBM's president and CEO, sent shock waves through the technology world in December when he announced the technology giant would invest \$1 billion in Linux in 2001 alone. He called Linux the best way to meet demands on the Internet.

IBM's track record bodes well for the future of Linux. The personal computer had been around for almost ten years before IBM, in 1983, introduced a PC that revolutionized the technology. The rest, as they say, is history.

IBM stated its belief in Linux as early as January 10, 1999, when it proclaimed in a *New York Times* story that Linux would be the third wave in computer technology, following the PC and the Internet.

IBM is not the only company out there using Linux. Others include Sony Electronics, ZDNet, Sallie Mae, Boeing and Compaq, just to name a few. Linux will not be a one-size-fits-all approach for every company's technology needs. But nearly all companies will benefit by using Linux in certain applications.

Sair Linux and GNU Certification

Sair Linux and GNU Certification, in our humble and somewhat partisan opinion, offers the best of all worlds. Sair Linux and GNU Certification has grown from a college student's simple question to his instructor (that would be me) just three years ago—"Is there a Linux certificate available?"—to a growing 100-employee company attracting students globally.

Sair Linux and GNU Certification provides multilayered, vendor-neutral training that has, in a few short years, produced 1,000 certified professionals and an additional 200 Linux certified administrators. Our training is administered through accredited centers of education (ACEs), which includes New Horizon Learning Centers and Productivity Point International training centers. Testing is offered through Thomson Prometric and VUE testing centers, and self-study guides are available through John Wiley & Sons Publishing.

Being vendor-neutral allows us to offer comprehensive training by offering instruction in a minimum of seven different Linux versions. We believe certification tied to a specific vendor will ultimately succeed or fail depending on that vendor's success.

Sair Linux and GNU Certification differs from other Linux certification efforts in other ways. SLGC does not publish standalone tests. Instead, Sair Linux and GNU Certification provides a knowledge array with a detailed set of objectives and competencies. This allows the Sair Linux and GNU Certification curriculum to be more focused and written to offer the best opportunity to pass the tests. The detailed knowledge array requirements also provide the necessary guidelines for others to write study guides for our exams.

Our results are not based on "psychometric" normative data. Instead, results are absolute; each exam contains 50 questions and 37 (or 74%) must be answered correctly for a passing grade. It does not matter what Joe or Hildegard scored on the test, all that matters is that you demonstrate mastery of the material by scoring 74% or better.

Level I provides a lengthy peer-reviewed and solid foundation of Linux education and contains four exams. You earn the Sair Linux and GNU Certified Professional (LCP) designation after passing either of the two initial exams: Installation and Configuration or System Administration. The Sair Linux and GNU Certified Administrator (LCA) designation is earned upon successful

completion of the remaining two Level I exams: Networking and Security, and Ethics and Privacy.

The recently announced Level II curriculum differs slightly. Where Level I is considered more “academic”, Level II focuses on various applications and practical uses.

The Sair Linux and GNU Certified Engineer (LCE) designation is earned by passing the Core Concepts and Practices (CCP) course and three elective courses. The curriculum is known as AMPS, an acronym derived from the many applications that run on Linux. Currently there are seven electives to choose from with more to be unveiled this year. AMPS allows candidates to select the applications that match their job descriptions. Do you want to be a web service provider? Or do you want to be a system administrator or work in a highly secured environment? You can custom-tailor your courseware based on how you plan to use Linux and GNU software.

A third level, that of Linux Certified Master Engineer, will be announced in 2002.

Naysayers may continue to look down on certification and training. And if they want to think certification is silly, that's fine. They can sit in their ivory towers and program. But, to paraphrase Shakespeare, “Wherefore art thou?”



Tobin Maginnis is the president and chief executive officer of Sair Linux and GNU Certification. He also is an associate professor of computer science at the University of Mississippi. He can be reached at ptm@sairinc.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Using Python to Query MySQL over the Net

Mihai Bisca

Issue #85, May 2001

Mihai shows how Python can be used to create a CGI script to enable elegant searching.

Recently, I became the owner of more than a thousand records of Go (an ancient oriental strategic board game) games played by professional or amateur players. All the games were stored in Smart Game Format (SGF), which is a text-based format designed to keep records of board games for two players. Naturally, I made them available to other players through my web page.

To make searching through the archive easier, I first used an HTML form with only one text input field. The user could enter a string (for example, a player name) that was passed to a Python CGI script which, in turn, invoked good old **grep** to find the matching files, as shown in Figure 1. But this is a crude way of finding information. For instance, it can be used to find all the games played by a certain player but not to find more complicated things, such as all the games that player played with black stones or all the games won by that player in 1995.

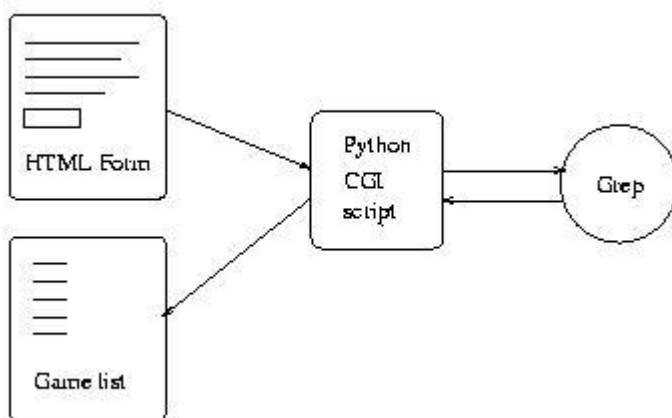


Figure 1. The Old, Crude Searching Method

In order to make a better search engine, I had to use a different approach. First, the database should describe the collection of game files. Then, a multiple input field form should be created to allow the user to search for various pieces of information at the same time. Finally, there should be a way to set up communication between the browser and database server, in order to make the result of the database query available to the user as an HTML document. Thus, the whole search would be performed as mapped in Figure 2.

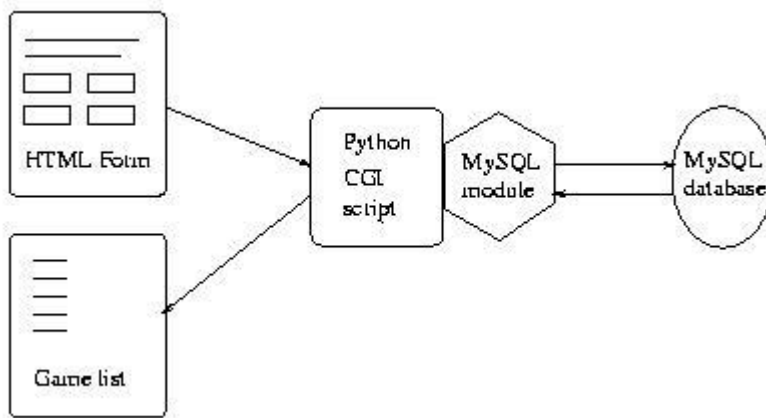


Figure 2. A More Elegant Search Method

The Tools

I chose MySQL 3.22.32 as the database and Python 1.5.2 for CGI scripting. I have played with Perl but prefer the feel of Python. The installation procedure for these programs has been explained already in many articles and will not be covered here. The reader should check the web sites in Resources for further details concerning installation. Communication between MySQL and Python is handled by a contributed module, presented below.

MySQLmodule

Python makes MySQL queries through a special module designed by Joerg Senekowitsch. Of course, several other modules are available on the Net, but MySQLmodule1.4 was easy to install and learn, and it worked very well for me. On some systems (like FreeBSD) it is possible to install this module at the same time as Python. On my Slackware 7.1 Linux, I had to build and install it as a dynamically loadable module. This is a three-step procedure: untar the MySQLmodule archive, compile the shared module and install the module somewhere in Python's library path.

Step one is fairly simple. As root, one would type:

```
myhost:~# tar xvzf MySQLmodule-1.4.tar.gz
```

A new directory named MySQLmodule-1.4/ will be created with several files, the most important of which are MySQLmodule.c (the source for the module to be compiled) and README (a file with installation and use information).

There are several tricky things about step two. For instance, one must know precisely where the libraries and include files for MySQL and Python can be found. On my system, MySQL 3.22.32 places the mysqlclient library in /usr/lib/mysql and the mysql.h include file in /usr/include/mysql. Python libraries can be found in /usr/lib/python1.5/config and the include file in /usr/include/python1.5. The command to compile MySQLmodule is:

```
myhost:~# gcc -shared -I/usr/include/mysql MySQLmodule.so
```

Another hint: the order of items in the above command line is important and must not be changed! Believe me, this is a hard-learned truth.

Step three consists of copying MySQLmodule.so to a directory where it can be found by Python at runtime. For Python 1.5 this could be /usr/lib/python1.5/lib-dynload, where other shared object files also reside. With Python 2.0 (which I also tested) I would recommend using the directory /usr/lib/python2.0/site-packages/.

Once the module is installed, it should be available from Python. It is a good idea to check this right away with a simple import statement such as:

```
myhost:~$ python
Python 1.5.2 (#1, May 28 2000, 18:04:10)
Copyright 1991-1995 Stichting Matematicisch Centrum,
Amsterdam
>>> import MySQL
>>>
```

If Python doesn't complain with an error message, chances are the MySQL module is properly installed and working.

The HTML Form

The user should be able to search the database for several items like the tournament name, the black and white player names, the date of the game (at least the year) and also the winner of the game. All this information is available in the SGF files along with the actual game record.

So, I wrote the HTML document shown in Listing 1. Yes, I like to write HTML by hand, and there is no need for anything more complicated. Of course, "myhost" has to be replaced with the actual hostname of the web server. The reader will also notice the HTML form acts by calling the CGI script named search.py once the submit button is clicked. The appearance of the document loaded by Netscape is shown in Figure 3.

Listing 1. Archive Search

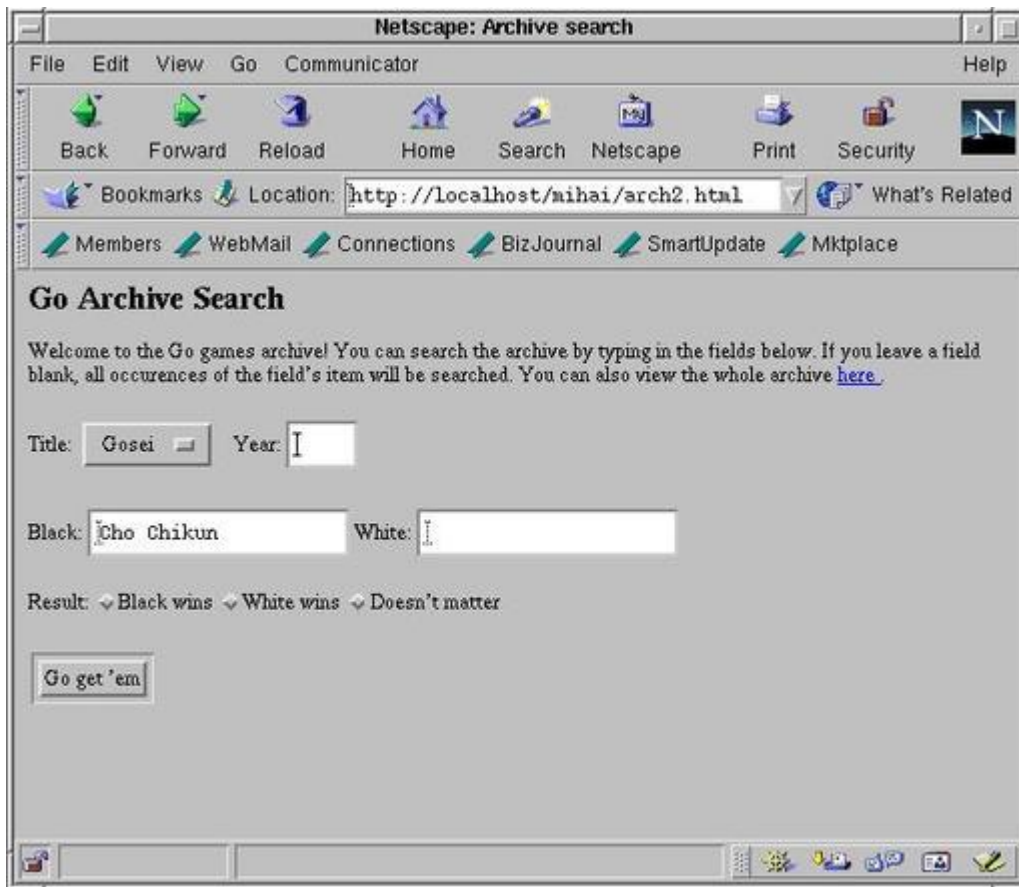


Figure 3. Document as Loaded in Netscape

The Igo Database

Each item in the HTML form has to be described in the database. To this end, I created a new database to store the tables. This is done as root, by typing:

```
myhost:~# mysqladmin create igo
```

where igo is the new database name. This is not enough though, because only root will have access to the new database. To grant only SELECT privileges to all users for the new database, root has to type:

```
myhost:~# mysql mysql
mysql> insert into db values (
  '%', 'igo', '', 'Y', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
  'N', 'N');
```

This changes the table database in the MySQL internal database by adding an entry for the igo database. Then, the MySQL daemon has to be restarted or the command **mysql> flush privileges;** has to be typed so that the MySQL daemon becomes aware of the privilege change.

Now all users should be able to access data in database igo, but only root can change the data.

The MySQL Tables

Creating the tables (one for each tournament) was easy. For instance, a table named gosei, for the tournament Gosei, was made with the following commands:

```
myhost:~# mysql igo
mysql> create table gosei (
    black varchar(30),
    white varchar(30),
    dt date,
    rez varchar(30),
    fname varchar(30),
    ;
Query OK, 0 rows affected, (0.00 sec)
mysql>
```

The table has five columns: black player's name, white player's name, the date of the game, the result and finally, the corresponding SGF file name.

Loading the data in the table is another matter. I suppose one way would be to type:

```
mysql> insert into gosei values ('Cho Chikun', 'Kato Masao', '1987-07-03', 'B+3.5', 'gosei87_1.sgf' );
```

but I'd rather learn a new programming language than enter that a thousand times. Fortunately, there is another way of loading data in a MySQL table—from a text file. Each row in the file matches a row in the table, and the fields are separated by white spaces, as shown in Table 1.

Table 1. Text to Be Loaded to MySQL Table

Suppose this file is also named gosei. To pass the data to a MySQL table, one would write

```
mysql> load data infile "gosei" into table gosei;
```

Then a query result should look like Table 2.

Table 2. Loading from a Text File

The reader might ask “Okay, but isn't making a huge text file another burden?” Actually, that's an easy task for yet another small Python script (that I will not show, because it's not directly related to our topic).

With the igo database created and the tables loaded with data, there was only one thing left to be done: write the Python CGI script that would take input

from the user via the HTML form, query the database and produce a list of matching game files.

The Python CGI Script

This program, named `search.py` is presented in Listing 2 [at the [L/ftp site](#)]. It makes use of two great modules imported in the third and fourth lines. The CGI module is almost like magic: it just gets the data submitted by the HTML form as a Python dictionary. The programmer does not need to be concerned with details like the method (GET or POST) used to send form data to the CGI script. Don't you love Python?

The MySQL module is also easy to use. With four simple statements, it opens the connection to the desired database, sends the query, gets the results and stores them in a list of row lists (a Python list whose members each contain one row).

The script has the following structure: getting the HTML form data, creating the database query string according to the form data and querying a table and printing the matching results.

Since there is one for each Go tournament, step three is repeated through a for loop as many times as necessary. The Python code and the appended comments are rather self-explanatory, so I will only comment on the lines:

```
print '<li><a href="http://myhost' + \
      '/cgi-bin/getsgf.py?file=' + e[0] + '">'
print e[0] + '</a>'
```

Here, `e[0]` is an SGF file name. Instead of merely printing the filename (which would be of limited help to the user), the print statements create an HTML anchor that sends the filename to the CGI script `getsgf.py`. This last script (which for the sake of simplicity will not be shown here) searches for the actual path to the SGF file and starts a Java applet to display the contents of the file in a nice, graphical way.

Of course, both `search.py` and `getsgf.py` must be made executable and moved to the `cgi-bin` directory. Also, a final trap to avoid: all the SGF game files must be placed somewhere within the `DocumentRoot` directory (as defined in `httpd` configuration files) in order to be found by a CGI script.

The result of running `search.py` is displayed in Figure 4 as seen by the user. Clicking on each filename starts the applet shown in Figure 5, which gradually displays the moves of the game.

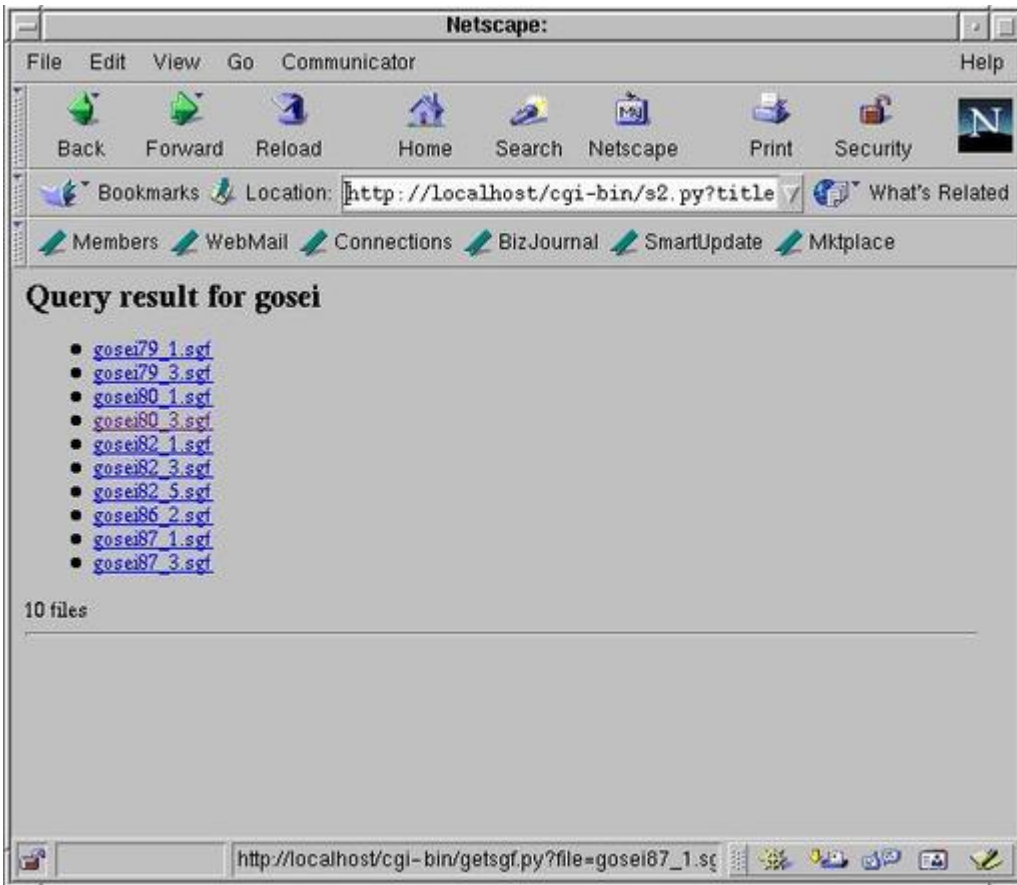


Figure 4. search.py Results

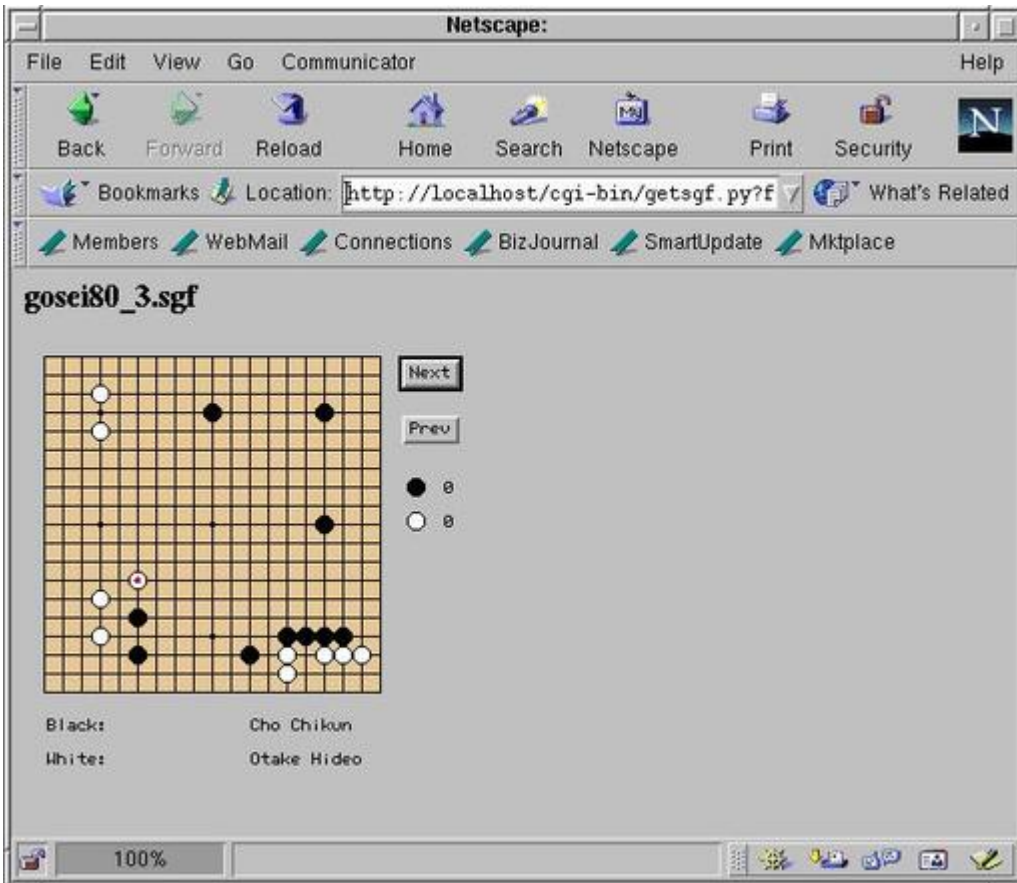


Figure 5. gose:80_3.sgf Applet

Conclusion

Python is a wonderful language. The MySQL module makes it easy to create small programs to retrieve data from a MySQL database. Python is also great for CGI scripting. Thus, having a database available to study through the web browser is just several lines of code away.

The application I have shown here is rather limited. The user can only search for five variables and in a fixed way, predetermined by the HTML form. However, it is conceivable for the user to write his or her own database queries through a textarea input field and view the query results on-line. In fact, the possibilities are limited only by the programmer's imagination.

Resources



Mihai Bisca, (AKA 5dan), is a former winner of the Romanian Go Championship. He spends countless hours playing with his Slackware Linux and dreams of working over the Internet from home. In his other, daytime life, he works as an ophthalmologist.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

MOSIX: A Cluster Load-Balancing Solution for Linux

Ibrahim F. Haddad

Evangeline Paquin

Issue #85, May 2001

Ibrahim introduces the MOSIX software package and describes how it was installed on an experimental Linux cluster in the Ericsson Systems Research Lab in Montréal.

Software clustering technologies have been evolving for the past few years and are currently gaining a lot of momentum for several reasons. These reasons include the benefits of deploying commodity, off-the-shelf hardware (high-power PCs at low prices), using inexpensive high-speed networking such as fast Ethernet, as well as the resulting benefits of using Linux. Linux appears to be an excellent choice for its robust kernel, the flexibility it offers, the various networking features it supports and the early availability of its IP releases.

ARIES Project

MOSIX

The ARIES Project

With the growth of the popularity of clustering technologies, we decided to start a project that aims at finding and prototyping the necessary technology to prove the feasibility of a clustered Linux internet server that demonstrates telecom-grade characteristics. Thus, a star was born, and it was called ARIES (advanced research on internet e-servers).

ARIES started at the Ericsson Core Unit of Research in January 2000, and its objective was to use the Linux kernel as the base technology and to rely on open-source software to build the desirable system with characteristics that include guaranteed availability and response time, linear scalability, high

performance and the capability of maintaining the system without any impacts on its availability.

Traffic distribution and load balancing were two main areas of investigation. The strategy followed was to survey the open-source world, check the available solutions in those areas, test them and determine to what extent they meet our requirements for the targeted near telecom-grade Linux internet server.

This article covers our experience with MOSIX, a software package developed at the Hebrew University of Jerusalem. We expose the MOSIX technology, the algorithms developed and how they operate and describe how we installed MOSIX on an experimental Linux cluster. We also discuss MOSIX's strengths and weaknesses in order to help others decide if MOSIX is right for them.

The MOSIX Technology

MOSIX is a software package for Linux that transforms independent Linux machines into a cluster that works like a single system and performs load balancing for a particular process across the nodes of the cluster. MOSIX was designed to enhance the Linux kernel with cluster computing capabilities and to provide means for efficient management of the cluster-wide resources. It consists of kernel-level, adaptive resource sharing algorithms that are geared for high performance, overhead free scalability and ease of use of a scalable computing cluster.

The core of MOSIX is its capability to make multiple server nodes work cooperatively as if part of a single system. MOSIX's algorithms are designed to respond to variations in the resource usage among the nodes by migrating processes from one node to another, pre-emptively and transparently, for load balancing and to prevent memory exhaustion at any node. By doing so, MOSIX improves the overall performance by dynamically distributing and redistributing the workload and the resources among the nodes in the cluster.

MOSIX Operation

MOSIX operates transparently to the applications and allows the execution of sequential and parallel applications without regard for where the processes are running or what other cluster users are doing.

Shortly after the creation of a new process, MOSIX attempts to assign it to the best available node at that time. MOSIX then continues to monitor the new process, as well as all the other processes, and will move it among the nodes to maximize the overall performance. This is done without changing the Linux interface, and users can continue to see (and control) their processes as if they run on their local node.

Users can monitor the process migration and memory usages on the nodes using QPS, a contributed program that is not part of MOSIX but supports MOSIX-specific fields. QPS is a visual process manager, an X11 version of “top” and “ps” that displays processes in a window and allows the user to sort and manipulate them. It supports special fields (see Figure 1) such as on which node a process was started, on which node it is currently running, the percentage of memory it is using and its current working directory.

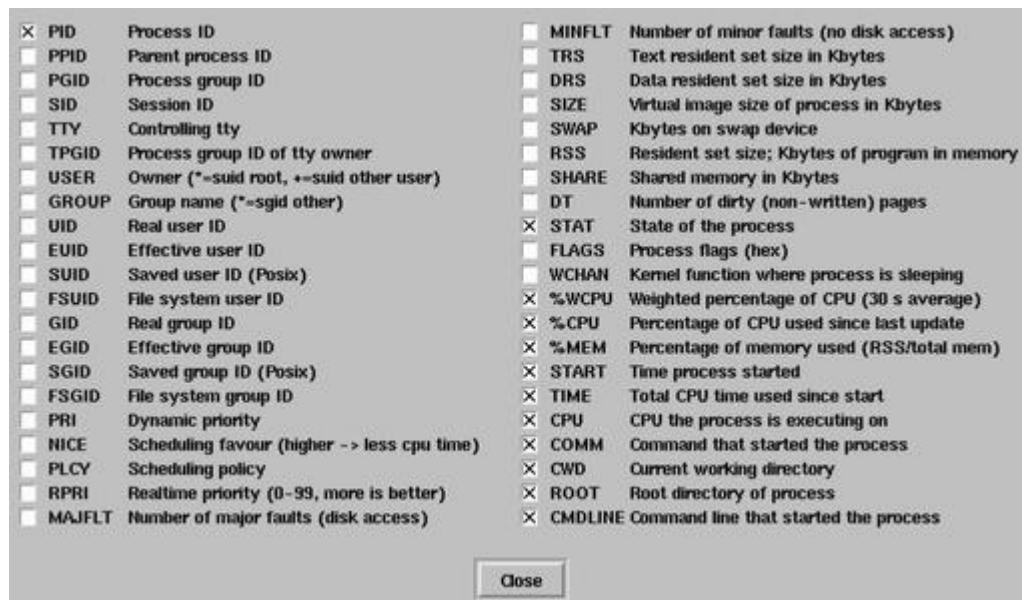


Figure 1. QPS Supported Fields

Since MOSIX's algorithms are decentralized, each node is both a master for processes that were created locally and a server for processes that migrated from other nodes. This implies that nodes can be added or removed from the cluster at any time, with minimal disturbances to the running processes.

MOSIX improves the overall performance by better utilizing the network-wide resources and by making the system easier to use. MOSIX's system image model is based on the home-node model. In this model, all the user's processes seem to run at the user's login-node. Each new process is created at the same node(s) as its parent process. Processes that have migrated interact with the user's environment through the user's home-node, but where possible, they use local resources. As long as the load of the user's login-node remains below a threshold value, all the user's processes are confined to that node. However, when this load rises above a threshold value, some processes may be migrated (transparently) to other nodes.

Scheduling Policy

MOSIX schedules newly started programs in the node with the lowest current load. However, if the machine with the lowest load level announces itself to all the nodes in the cluster, then all the nodes will migrate newly started jobs to

the node with the lowest load and soon enough this node will be overloaded. However, MOSIX does not operate in this manner. Instead, every node sends its current load status to a random list of nodes. This prevents a single node from being seen by all other nodes as the least busy and prevents any node from being overloaded.

How does MOSIX decide which node is the least busy among all the cluster nodes? This is a good question; however, the answer is a simple one.

MOSIX comes with its own monitoring algorithms that detect the speed of each node, its used and free memory, as well as IPC and I/O rates of each process. This information is used to make near optimal decisions on where to place the processes. The algorithms are very interesting because they try to reconcile different resources (bandwidth, memory and CPU cycles) based on economic principles and competitive analysis. Using this strategy, MOSIX converts the total usage of several heterogeneous resources, such as memory and CPU, into a single homogeneous cost. Jobs are then assigned to the machine where they have the lowest cost. This strategy provides a unified algorithm framework for allocation of computation, communication, memory and I/O resources. It also allows the development of near-optimal on-line algorithms for allocation and sharing these resources.

MOSIX Filesystem (MFS)

MOSIX uses its own filesystem, MFS, to make all the directories and regular files throughout a MOSIX cluster available from all nodes as if they were within a single filesystem. One of the advantages of MFS is that it provides cache consistency for files viewed from different nodes by maintaining one cache at the server disk node.

MFS meets the direct file system access (DFSAs) standards, which extends the capability of a migrated process to perform some I/O operations locally, in the current node. This provision reduces the need of I/O-bound processes to communicate with their home-node, thus allowing such processes (as well as mixed I/O and CPU processes) to migrate more freely among the cluster's node, for load balancing and parallel file and I/O operations. This also allows parallel file access by proper distribution of files, where each process migrates to the node that has its files.

By meeting the DFSAs provision, allowing the execution of system calls locally in the process' current node, MFS reduces the extra overhead of executing I/O-oriented system calls of a migrated process.

Installation Environment

In order to test MOSIX, we set up the following environment: 1) a cabinet that consists of 13 Pentium-class CPU cards running at 233MHz with 256MB of RAM each; and 2) a Pentium-based server machine, PC1, running at 233MHz with 256MB of RAM. This machine was used as an NFS and DHCP/TFTP server for the 13 diskless CPUs.

When we start the CPUs, they boot from LAN and broadcast a DHCP request to all addresses on the network. PC1, the DHCP server, will be listening and will reply with a DHCP offer and will send the CPUs the information needed to configure network settings such as the IP addresses (one for each interface, eth0 and eth1), gateway, netmask, domain name, the IP address of the boot server (PC1) and the name of the boot file. The CPUs will then download and boot the specified boot file in the DHCP configuration file, which is a kernel image located under the /tftpboot directory on PC1. Next, the CPUs will download a ramdisk and start three web servers (Apache, Jigsaw and TomCat) and two streaming servers (Real System Server and IceCast Internet Radio).

Installation Steps

For this setup, we used the Linux Kernel 2.2.14-5.0 that came with Red Hat 6.2. At the time we conducted this activity, MOSIX was not available for Red Hat; thus, we had to port MOSIX to work with the Red Hat kernel. Our plan was to prepare a MOSIX cluster that consists of the server, PC1 and the 13 diskless CPUs. For this reason, we needed to have a MOSIX-enabled kernel on the server, and we wanted to have the same MOSIX-enabled kernel image under the TFTP server directory to be downloaded and started by the CPUs at boot time. After porting MOSIX to Red Hat, we started the MOSIX modified installation script "mosix.install" that applied the patches to the 2.2.14-5.0 kernel tree on PC1.

Once we finished configuring the kernel and enabling the MOSIX features (using `$make xconfig`), we compiled it to get a kernel image:

```
cd /usr/src/linux
make clean ; make dep ;
modules_install
```

Next, we copied the new kernel image from /usr/src/linux/arch/i386/boot to /boot and we updated the System.map file:

```
cp /usr/src/linux/arch/i386/boot/bzImage
cp /usr/src/linux/arch/i386/boot/System.map
ln /boot/System.map.mosix /boot/System.map
```

One of the configuration files that was modified was lilo.conf. We added a new entry for the MOSIX kernel to make the server boot as a MOSIX node by default. The updated lilo.conf on PC1 looked like Listing 1.

Listing 1. A MOSIX-Modified lilo.conf.

Having done that, we needed to complete the configuration steps. In /etc/profile, we added one line to specify the number of nodes in the MOSIX cluster:

```
# Add to /etc/profile NODES=1-14
```

We created /etc/mosix.map that allows the local MOSIX node to see all other MOSIX nodes. The mosix.map looked as follows:

```
# Starting node  IP          Number of Nodes
1                x.x.x.x      13
14               y.y.y.y       1
```

We created the /mfs directory to be used as a mount point for the MOSIX filesystem. We added mosix.o to /lib/modules/2.2.14-5.0/misc/ so it can be loaded at boot time by the MOSIX startup file. Then we applied the same modifications to the ramdisk that will be downloaded by the diskless CPUs at boot time.

Once we completed these steps, we rebooted PC1, and when it was up and running, we rebooted the diskless CPUs. After reboot, the diskless CPUs received their IP addresses, booted with the MOSIX-enabled kernel, and downloaded the ramdisk using the TFTP protocol. *Et voilà!* All 14 nodes mounted /mfs as the MOSIX filesystem directory. Figure 2 shows a snapshot of /mfs on CPU10.

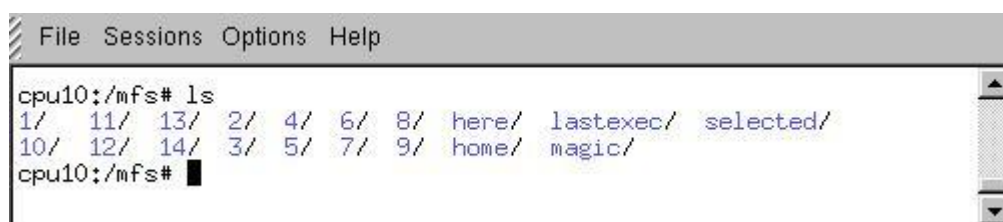


Figure 2. MFS Mount Point on CPU10

Testing the Installation

After starting the 14 nodes as a MOSIX cluster, we wanted to test our installation. By default, all the diskless CPUs mount an NFS directory on PC1. So we placed the Linux kernel 2.2.14 source code directory under that NFS space, making it visible to all nodes, and we started the kernel compilation process using **MExec/MPMake**, the parallel make, a contributed software that assigns

new processes to the best available cluster nodes (available for download from MOSIX web site).

Figures 3, 4, 5 and 6 show snapshots of **mon**, a MOSIX tool that shows the load on all the nodes. As Figure 3 shows, there was a high load on node 14 because it was the node on which the compilation started. A few seconds later, Figures 4 and 5 show less load on CPU 14, and then Figure 6 shows a good distribution of the load among all the nodes.

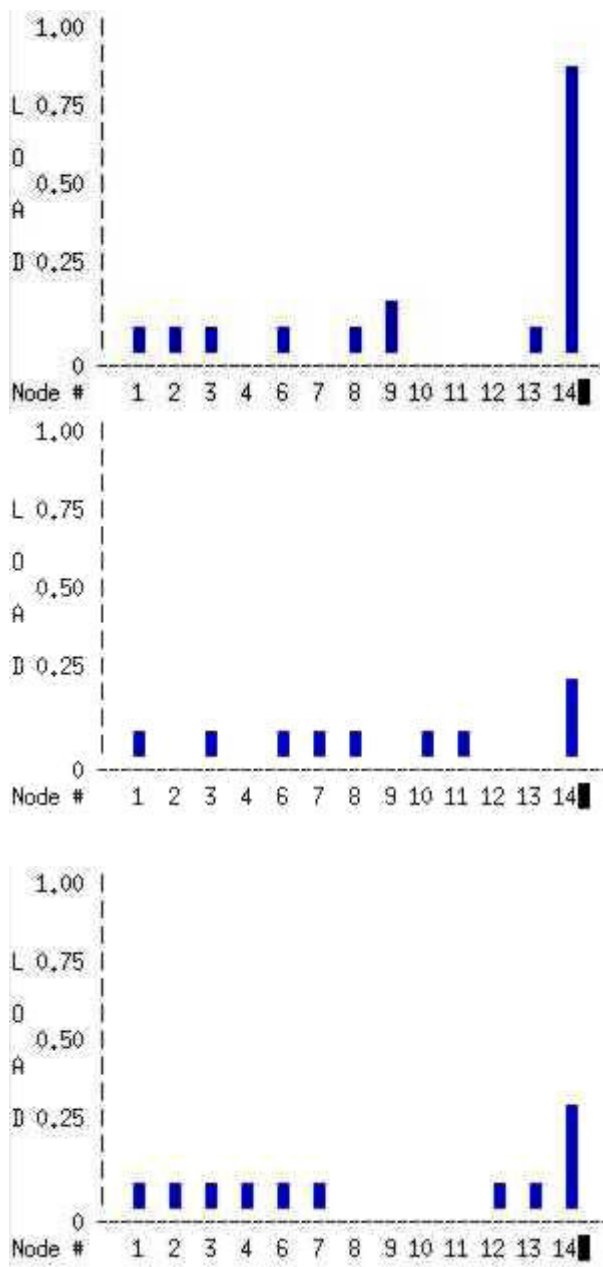
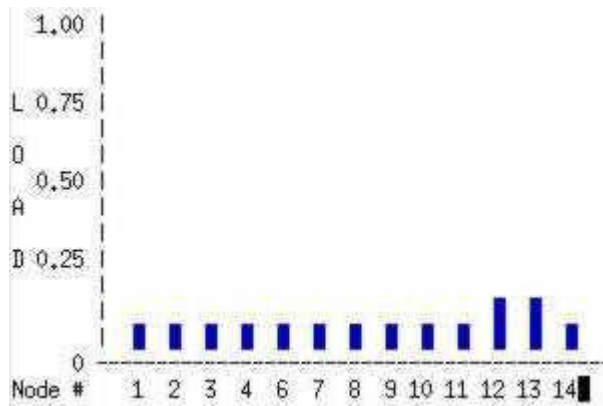


Figure 3, 4, 5 and 6. MOSIX at Work Distributing Loads



Scalability

MOSIX supports configurations with large numbers of computers with minimal scaling overheads to impair the performance. You can have a simple low-end setup composed of several PCs connected via Ethernet, on the other hand, you can have larger configurations that include workstations and servers connected via higher speed LANs such as fast Ethernet. A high-end configuration may also include a large number of SMP and non-SMP workstations and servers connected via a high-performance LAN such as Gigabit-Ethernet.

Our last experiment will include testing MOSIX on a new self-contained NEBS-compliant cabinet that consists of 16 Pentium III processors powered with 512MB of RAM and running at 500MHz. Each CPU has two on-board Ethernet ports and is also paired with a four-port ZNYX Ethernet card (used to provide Ethernet redundancy). Eight of the CPUs have a RAID setup (RAID 0 and RAID 5) with three 18GB SCSI disks.

Getting MOSIX

MOSIX for Linux is subject to the GNU General Public License version 2, as published by the Free Software Foundation. It is available for download from the MOSIX web site (see Resources).

Uninstalling MOSIX

MOSIX allows us to do an uninstall and clean up the kernel source it modified. During the initial installation, mosix.install modifies the following system configuration files: /etc/inittab, /etc/inetd.conf, /etc/lilo.conf, /etc/rc.d/init.d/atd and /etc/cron.daily/slocate.cron.

The original contents of these files are saved with the .pre_mosix extension and the changes made to kernel files are logged to the mos_uninstall.log file in the kernel-source directory. To uninstall MOSIX, you run the command `./mosix.install uninstall` and answer the questions. When you are asked if you want to clean the Linux kernel sources, answer "yes". The script will then

attempt to revert all the changes that were made during a previous MOSIX installation. At the end you need to reboot the node so start it as a plain Linux node.

Conclusion

Clustering offers several advantages that result in sharing the processing and the ability to achieve higher performance. If you are interested in clustering your servers with efficient load-balancing software and you need support for high performance, then MOSIX can certainly be useful for you. It is easy to install and configure, and it works.

However, our initial interest with MOSIX was to understand its algorithms and investigate the possibility of using it for efficient distribution of web traffic over multiple processors. We found that MOSIX is not directly suitable for the type of functionality we want for a near telecom-internet server that we are aiming to prototype, mainly because it is missing a front-end tool for transaction-oriented load balancing such as the HTTP requests.

There have been many requests to the MOSIX mailing list asking about HTTP traffic distribution with MOSIX. I believe that if the authors would add this functionality to MOSIX, MOSIX will be one of the most popular software packages for Linux clusters.

Acknowledgements

The Systems Research Department at Ericsson Research Canada for providing the facilities and equipment as well as approving the publication of this article. Marc Chatel, Ericsson Research Canada, for his help and support in the lab.

Resources



Ibrahim F. Haddad (ibrahim.haddad@lmc.ericsson.se) works for Ericsson Research Canada in the Systems Research Department researching carrier class server nodes in real time on IP networks. He is currently a DrSc Candidate in the Computer Science Department at Concordia University.

Evangeline Paquin (lmcevpa@lmc.ericsson.se) is a computer science student at UQAM University in Montréal. She completed her coop training at Ericsson

Research Canada working on Linux clustering solutions, and currently she is a part-time staff member in the System Research Department.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Apache Toolbox

Ralph Krause

Issue #85, May 2001

The Apache Toolbox simplifies installation and configuration of the Apache web server and related modules.

The Apache web server is one of the most common applications run on Linux systems. Although it comes with most distributions there will be times when you want to run the latest version along with all of its modules. Getting all the source code and then configuring, compiling and installing each piece by hand can be a daunting task. The Apache Toolbox provides an easier way to do all of this.

The Apache Toolbox, written by Bryan Andrews, provides a convenient front end for configuring and compiling Apache, as well as obtaining any needed source code. In addition to the modules included with Apache, the Toolbox can also configure and compile programs such as PHP and MySQL (See Table 1 for a complete list of software). It also compiles and installs the latest gd libraries for the creation of JPEG and PNG files.

[Table 1. Modules Supported by the Apache Toolbox](#)

Obtaining and Installing the Apache Toolbox

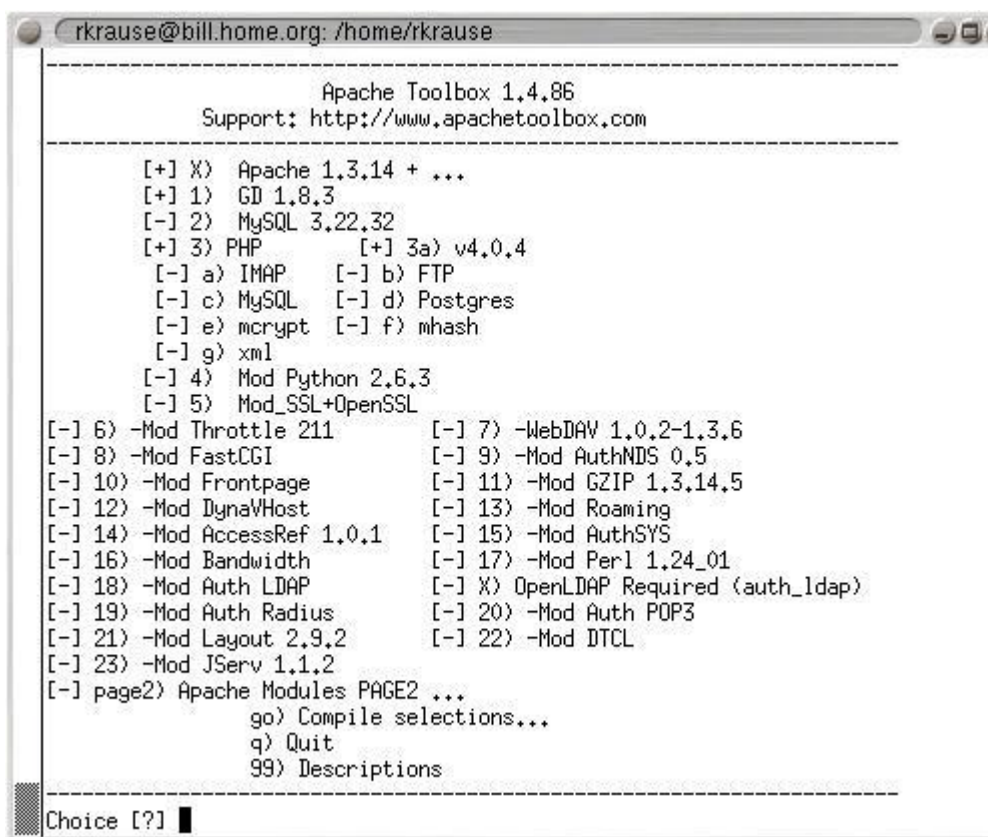
The Apache Toolbox comes in two forms: a large (approximately 12MB) file and a small file. The large file contains the Toolbox script along with source code for Apache and most of the modules; the small file contains only the Apache Toolbox script. Since the Apache Toolbox tries to download any missing source code, it makes sense to use the smaller file if you have a recent Linux distribution or a fast internet connection.

If you download the large Apache Toolbox tarball it will untar into a directory called www-src. This directory contains zipped tarballs of the source code for

Apache and all the modules. The Apache Toolbox script, called `install.sh`, is also in this directory.

Running the Apache Toolbox

To start the Apache Toolbox, switch to root and run the `install.sh` script. This presents a menu of programs and modules that can be selected (see Figure 1). A selection is made by typing the number or letter next to an item. The program performs sanity checks, depending upon your selection. It ensures, for example, that Python is installed on the system when the `mod_python` module is selected. The program also displays descriptions of the modules when "99" is entered. There are two pages of modules, and the 99 command only displays descriptions for the current page.



```
rkrause@bill.home.org: /home/rkrause
-----
                Apache Toolbox 1.4.86
                Support: http://www.apachetoolbox.com
-----

[+] X) Apache 1.3.14 + ...
[+] 1) GD 1.8.3
[-] 2) MySQL 3.22.32
[+] 3) PHP          [+] 3a) v4.0.4
    [-] a) IMAP      [-] b) FTP
    [-] c) MySQL     [-] d) Postgres
    [-] e) mcrypt    [-] f) mhash
    [-] g) xml
[-] 4) Mod Python 2.6.3
[-] 5) Mod_SSL+OpenSSL
[-] 6) -Mod Throttle 211      [-] 7) -WebDAV 1.0.2-1.3.6
[-] 8) -Mod FastCGI          [-] 9) -Mod AuthMDS 0.5
[-] 10) -Mod Frontpage       [-] 11) -Mod GZIP 1.3.14.5
[-] 12) -Mod DynaVHost       [-] 13) -Mod Roaming
[-] 14) -Mod AccessRef 1.0.1 [-] 15) -Mod AuthSYS
[-] 16) -Mod Bandwidth       [-] 17) -Mod Perl 1.24_01
[-] 18) -Mod Auth LDAP      [-] X) OpenLDAP Required (auth_ldap)
[-] 19) -Mod Auth Radius    [-] 20) -Mod Auth POP3
[-] 21) -Mod Layout 2.9.2   [-] 22) -Mod ITCL
[-] 23) -Mod JServ 1.1.2
[-] page2) Apache Modules PAGE2 ...
        go) Compile selections...
        q) Quit
        99) Descriptions
-----
Choice [?] █
```

Figure 1. The Apache Toolbox's Configure Script Menu

Once you have made all of your selections, type **go** to start the build process. The first thing that the Toolbox does is warn you about any installed RPMs that conflict with the software it will install. You are given the option to continue the compilation process or to quit to remove the offending packages. If you quit to remove the offending RPMs, the Apache Toolbox remembers your settings when you restart it.

The Apache Toolbox uses **wget** to download any packages that it needs. If `wget` isn't installed on your machine, the Toolbox can download it, using `Lynx`, and

install it for you. The FTP locations for all the modules are hard-coded in the `install.sh` script so you can verify them to ensure that you're getting legitimate files.

After the `wget` checks are performed, you are asked if you want to change Apache's default installation path. Following this question, the Apache tarball is uncompressed. If the program finds existing Apache source code in the `www-src` directory, you are given the option to back it up. Next, Apache is preconfigured based upon the selections that you have made.

Once that is done the selected modules are untarred, built and installed. If the script can't locate source code for a module, you are asked if it should be downloaded. The Apache Toolbox uses the `installwatch` library to log the results of the compile and the install process for each module to the logs directory.

If you have elected to install PHP, you are given the option of editing its configuration script. You can use the editor of your choice to edit the file, and the PHP build process continues when you are done.

Once all the modules have been compiled, you are given the option to modify the Apache configuration script. After that the Toolbox tailors the Apache Makefile for your module selections. Then you have to compile and install Apache by changing to the Apache source directory and typing **make**. If Apache compiles without error, then type **make install** to install it.

Caveats

Because the Apache Toolbox uses the latest versions of Apache, PHP, etc., it is important to have a fairly recent Linux distribution. The first machine I attempted to use the Apache Toolbox on was an older one running SuSE 6.1 and the `installwatch` library and PHP wouldn't compile on it. I moved to a machine running Red Hat 6.2, and everything compiled without incident.

I had originally downloaded the large Apache Toolbox tarball complete with source code. When a newer version came out I just downloaded the small Apache Toolbox file and put the newer `install.sh` script in my existing `www-src` directory. Even though the newer script was supposed to use an updated version of PHP, it kept using the one installed by the older Apache Toolbox. Once I deleted the configuration file (`config.cache`) created by the earlier version of the Apache Toolbox the newer version of PHP was used.

While the Apache Toolbox automates the configure, compile and install process, you might have to do a few things by hand. Examples of this include modifying your configuration files to start Apache or changing the default MySQL password.

The Apache Toolbox menu and description pages are a few lines longer than what fits on a normal display, so you might have to scroll a bit to read everything. I also noticed that descriptions for some of the modules, such as `mod_auth_radius` and `mod_auth_POP3`, were missing from the descriptions page.

Conclusion

The Apache Toolbox automates the process of obtaining and compiling Apache and Apache modules. It verifies that your system has the prerequisites for running modules and warns about installed RPMs that conflict with the software that it is installing. Its simple menu interface makes it easy to configure Apache to use a wide variety of modules in different combinations. While it doesn't automate the entire process, it does take care of the most tedious tasks.

Resources



Ralph Krause lives in Algonac, Michigan and has been using Linux for several years. He works as a writer and a web-master and can be reached at rkrause@netperson.net.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Boot with GRUB

Wayne Marshall

Issue #85, May 2001

Especially useful for multiboot, partitioned systems, GRUB offers flexibility and convenience for startup.

GRUB: it's neither larva, fast food nor the loveliest of acronyms in the GNU herd of free software. Rather, GRUB is the GNU GRand Unified Bootloader. And, it is truly the greatest loader for booting Linux and practically any other OS—open source or otherwise—you may have scattered on your platters.

GRUB is independent of any particular operating system and may be thought of as a tiny, function-specific OS. The purpose of the GRUB kernel is to recognize filesystems and load boot images, and it provides both menu-driven and command-line interfaces to perform these functions. The command-line interface in particular is quite flexible and powerful, with command history and completion features familiar to users of the bash shell.

GRUB is in its element with the multiboot, multidisk systems typical of Linux and open-source adventurers who may simultaneously test or track several Linux distributions, the BSDs, GNU/Hurd, BeOS and perhaps that vestigial partition for Mr. Bill. Even if you stick with LILO as your system's primary boot loader, it's smart to keep a GRUB boot floppy handy as the best and fastest way to get your system back if you otherwise cream your master boot record (MBR). If you have done any number of multiboot installations, you know exactly what I'm talking about. Should you need any more reasons for considering GRUB, check out the sidebar, "Why GRUB". Let's get started!

Installation

Installation of GRUB is a two-step process. The first step is to install or build GRUB in a host OS environment, and for this we will, of course, use Linux. The second step is to install and configure GRUB as the boot loader for your system.

The first step is the usual: download the source archive, untar it, configure and make install. Assuming you have found a source mirror (see www.gnu.org/software/grub/grub.html) and downloaded the source distribution into a suitable working directory, continue with:

```
tar -xzvf grub-0.5.96.1.tar.gz
cd grub-0.5.96.1
./configure
make
make install
```

This should create the executables: **grub**, **grub-install** and **mbchk**; install support files in `/usr/local/share/grub/i386-pc/`, and install the GNU information manual and man pages.

For the second step of installation, we will first build and work with a GRUB boot floppy. This way we can use GRUB to learn about its features while testing various configurations for our particular system. After getting comfortable with the GRUB setup on floppy, we will then install it onto the MBR of the system's first hard disk. Even if you decide not to install GRUB on your hard disk right away, no harm done: you will now have your own GRUB boot floppy available to rescue systems with trashed boot loaders.

Preparing a GRUB floppy

GRUB recognizes a number of different filesystem types, including Linux ext2fs, Reiser, MINIX, BSD's ffs, as well as FAT, so it is possible to make a GRUB boot floppy with any of these filesystems. We will stick to FAT for this example, however, because it is the lowest common denominator, and most OSes have tools for mounting and reading/writing files on FAT floppies. That way, we will always be able to get to its menu configuration file if we need to.

Scrounge around in your junk drawer for some unused floppy (a new one would be even better), and give it a fresh format and FAT filesystem:

```
fdformat /dev/fd0
mkfs -t msdos /dev/fd0
```

We are going to put some files on this disk, so go ahead and mount to your usual floppy mount point (here I use `/floppy`):

```
mount -t msdos /dev/fd0 /floppy
```

Now install the directories and files GRUB will need:

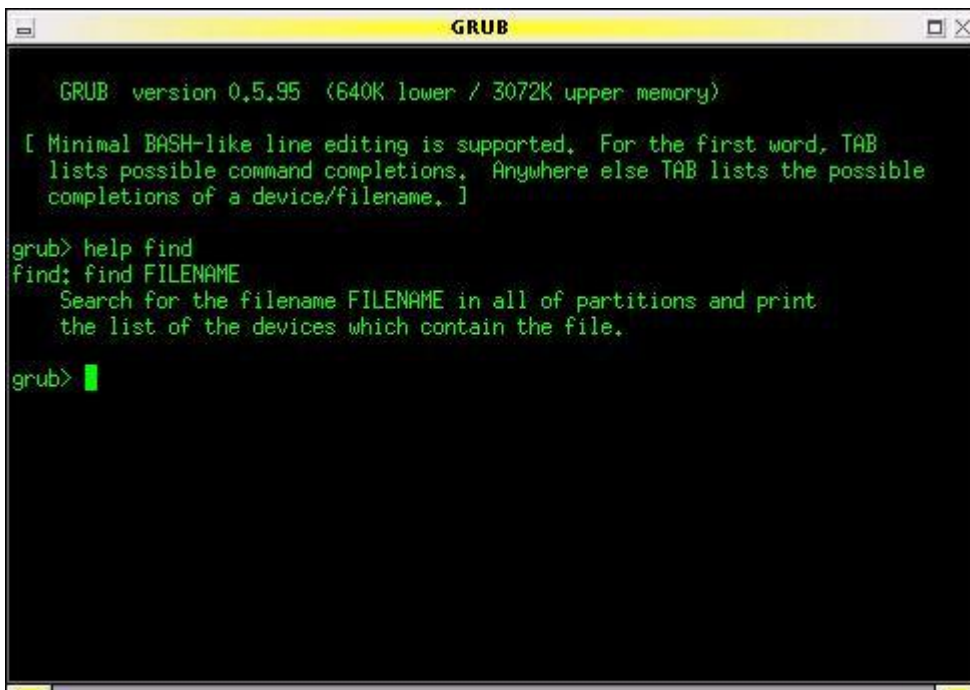
```
mkdir -p /floppy/boot/grub
cp /usr/local/share/grub/i386-pc/stage* /floppy/boot/grub
```

The floppy can then be unmounted, **umount /floppy**, but leave it in the drive. The GRUB floppy is prepared and ready for the final installation, which is to

install the GRUB boot loader in the MBR of the floppy itself. For that, we will use the grub executable we have built with our Linux installation. Start the executable at the Linux command prompt: **grub**.

This brings up an emulator of GRUB's command shell environment, which looks like Figure 1. We will discuss the features of this shell in more detail a little further on. For now, enter the following series of commands at the grub prompt:

```
grub> root (fd0)
grub> setup (fd0)
grub> quit
```



```
GRUB version 0.5.95 (640K lower / 3072K upper memory)

[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ]

grub> help find
find: find FILENAME
  Search for the filename FILENAME in all of partitions and print
  the list of the devices which contain the file.

grub> █
```

Figure 1. GRUB in command-line mode. Note the on-line help (here the GRUB emulator is running under Linux in an xterm window).

And that's it! This sequence of commands completes the installation of GRUB on the floppy disk. It is now bootable and will allow us to boot any other OS on our system.

Demonstrating GRUB

To see how GRUB may be used to boot a multitude of different operating systems, consider this example setup:

First Hard Disk (SCSI, Linux /dev/sda): 1st primary partition: Win98 2nd primary partition: Linux-Slackware 3rd primary partition: Linux-Debian 4th primary partition: Linux-Swap
Second Hard Disk (SCSI, Linux /dev/sdb) 1st primary partition: FreeBSD 2nd primary partition: OpenBSD 3rd primary partition: BeOS

Note that although GRUB and Linux are capable of dealing with installations in extended partitions, here we show a preference for using primary partitions whenever possible. Filesystems in primary partitions are often mountable by other operating systems, whereas cross-OS mounting filesystems in extended partitions is often not supported.

This system has two hard disks with six different operating systems using seven partitions. As you probably know, each OS has its own nomenclature for naming devices and partitions. For example, the Slackware installation would be known to Linux as `/dev/sda2` (with swap on `/dev/sda4`), while FreeBSD would recognize its filesystem on `/dev/da1s1a`. Alternatively, if the system were configured with IDE hard disks, Slackware would be on `/dev/hda2`, and FreeBSD would refer to its root directory on `/dev/ad1s1a`. You get large helpings of this alphabet soup whenever maintaining any multiboot setup.

Since GRUB also needs to be capable of loading any of these systems, it has its own OS-neutral naming conventions for referring to devices. Hard disks are all `hd`, floppy disks are `fd`, device numbering starts from zero, partition numbering starts from zero and complete device names are enclosed in parentheses.

With these naming rules, the floppy disk is `(fd0)`, the Win98 partition is `(hd0,0)`, and GRUB recognizes the Slackware and Debian partitions respectively as `(hd0,1)` for slackware and `(hd0,2)` for debian.

The BSDs further subdivide their own partitions (or “slices” in BSD terms), and GRUB would refer to the root mount for the FreeBSD system on `(hd1,0,a)`.

Okay, ready to give GRUB a taste? Slide the GRUB floppy in the drive and reboot your system (with your system's BIOS configured to boot from A: drive). You should see GRUB's terse boot messages and then find yourself in the GRUB command-line environment as shown in Figure 1.

To start, let's boot Slackware. Enter the following commands at the grub prompt:

```
grub> root (hd0,1)
grub> kernel /vmlinuz root=/dev/sda2 ro vga=791
grub> boot
```

Badda-bing, badda-boom, that postage-stamp-sized Tux appears in the upper-left corner of your screen (yes, Slackware is configured to use the framebuffer device), and Linux bootstraps its jolly way into glorious being.

Another example. Reboot the system again with the GRUB floppy, and enter the following commands at the grub prompt:

```
grub> rootnoverify (hd0,0)
grub> makeactive
grub> chainloader +1
grub> boot
```

Now your screen turns into a vague blue cloud, and you think you have made some horrible mistake. Then you realize it's only Windows and you remind yourself to expunge this partition one day soon.

Let's take a closer look at these examples. In the Slackware boot, we first used the GRUB **root** command to specify the device for GRUB to access. If the device has a filesystem recognized by GRUB (that is, one of ext2fs, reiser, ffs, etc.), it attempts to mount it and get its partition information, then reports its success following the command. Thus, you would see the following command/response dialog on your screen:

```
grub> root (hd0,1)
Filesystem type is ext2fs, partition type 0x83
```

Next, we used the GRUB **kernel** command to specify the boot image for GRUB to load. The argument to the kernel command is the filename of the boot image relative to the device specified by the root command above. The kernel image file can also be specified in explicit (device)/filename terms as follows:

```
grub> kernel (hd0,1)/vmlinuz
```

The kernel command gives you great flexibility for specifying the boot image you wish to load. For example, if we saved a previous version of a kernel to the file `/vmlinuz.old`, we could specify it with this command (which shows GRUB's response):

```
grub> kernel /vmlinuz.old root=/dev/sda2 ro vga=ask
[Linux-bzImage, setup=0xe00, size=0xfad30]
```

The arguments following the name of the boot image are passed to the target kernel and aren't related to GRUB. For Linux, these kernel arguments are pretty much what you would specify them to be in `lilo.conf`. In our example, we tell the kernel what device to mount for the root partition (`root=/dev/sda2 ro`), using the device nomenclature expected by Linux. Note here that we also use the **ro** flag to mount the root filesystem read-only initially while it performs its filesystem check. The other kernel argument in our example simply demonstrates setting another kernel variable (`vga=791`) to use a particular vga mode for the framebuffer display.

Finally, the last command is **grub> boot**. The kernel image specified is now loaded and sent rolling down the royal road to bootdom.

The second example, using Win98, demonstrates the use of GRUB's chain-loading mechanism. This method of booting loads the target OS's own boot-

chain-loader rather than a kernel image of the OS. In this instance, we specified:

```
grub> rootnoverify (hd0,0)
grub> chainloader +1
```

First, the **rootnoverify** command is for OS filesystems not specifically recognized by GRUB, so that GRUB will not try to mount the partition. Next, the chainloader command will use the first sector of the partition of device (hd0,0) and attempt to boot whatever it finds there. This is a common means of booting OSes that install their own boot loaders in the first sector of the partition where they are installed (this is sometimes called the partition boot sector or PBR).

Finally, the **makeactive** command sets the active flag in the partition table for the device specified by the root command, as some operating systems, like Win98, require.

The GRUB command line is easy and fun, and you should boot the different OSes on your system a few times to get the hang of it. While you are testing, be sure to keep any notes specific to getting your particular kernels successfully loaded. This information will be useful later when you configure the menu system of GRUB to perform these command-line steps automatically.

But before we leave the command line, here are a few more GRUB commands to look at.

The **help** command will display a list of the 40 or so commands available in GRUB. Typing the name a particular command after help will produce on-line help for that particular command. So **grub> help kernel** will tell you all about using the kernel command.

The **cat** command can be used to view the contents of a file. For example, **grub> cat (hd0,2)/etc/fstab** will show the contents of the /etc/fstab file in the Debian installation. This is a very handy way of pulling out system configuration information if your normal boot loader gets whacked. Note also as you are using the GRUB command line that, like bash, up and down arrows will scroll through command history, and a tab will complete the name of a GRUB command or filename.

Finally, you can call up a specific menu interface with the **configfile** command as in:

```
grub> configfile (fd0)/boot/grub/menu.lst
```

This will switch GRUB into its menu mode with an interface defined by the file, menu.lst. We haven't created that file yet, but—look out, segue coming!—that's exactly what we will do next.

Menu Configuration

Using the GRUB command line is cool, but after a few thousand system starts, you will probably get a little tired of entering the same commands at the GRUB prompt and long for something a little more automated. Good news from the GRUB gang: you get a fully configurable menu interface at no extra charge! The GRUB boot menu gives you point-and-shoot boot selection, unattended default boot after a configurable timeout, any number of fallback boots if previous boots fail, toggle between command-line and menu modes, and interactive editing of menu selections and password protection. These features give GRUB an ease of use to match its tremendous functionality.

When GRUB boots, it automatically looks for the /boot/grub/menu.lst file on its boot device (the last three letters are “ELL ess tee” and not “one ess tee”). If the file is found, GRUB automatically enters its menu mode and presents the user with a stunning interface, as shown in Figure 2.

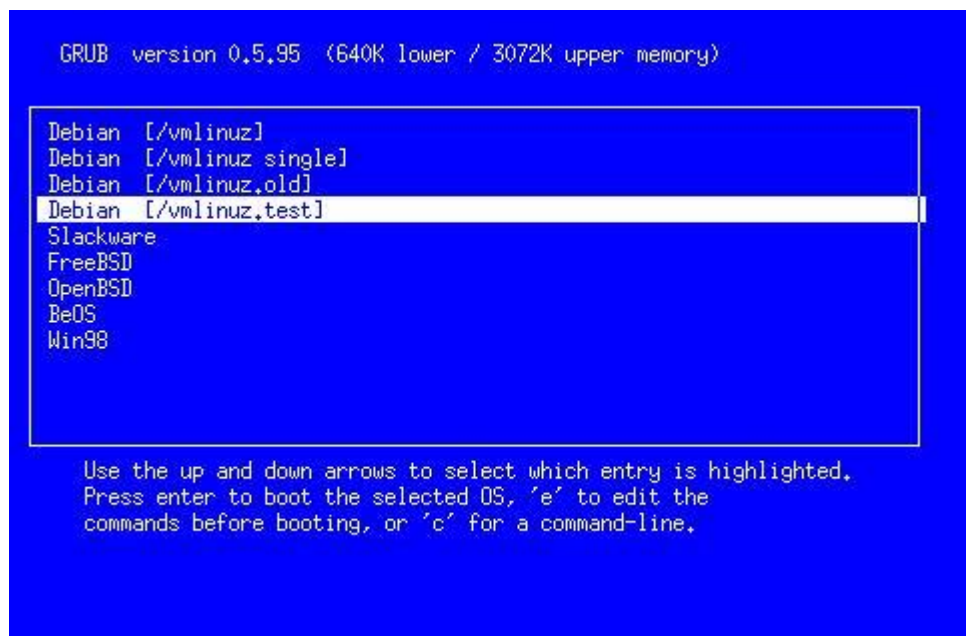


Figure 2. GRUB's Boot Menu Interface

Listing 1 [found at [L's web site](#)] shows the configuration file responsible for this demonstration menu. As you can see, it is a simple text file typical of many UNIX configuration files, where lines starting with hashes (#) and blank lines are ignored.

The first set of commands sets general configuration variables. The **timeout** command sets the time in seconds to wait for the user to make a selection

before proceeding automatically to the **default** boot. The default command sets which of the following boot stanzas GRUB should attempt to boot automatically. Boot stanzas are numbered implicitly, starting from zero, according to their order of appearance in the configuration file. This order is also how they will be listed in the menu.

The **fallback** command specifies which of the boot stanzas to load if the default should fail. It is possible to set more than one fallback, as is shown in the example.

The **color** command lets you breathe a bit of life into the text-mode menu screen. The syntax for the color command is

```
color foreground/background [ hilite-fg/hilite-bg ]
```

where each of the foreground/background colors is specified with a name from the set of black, blue, green, cyan, red, magenta, brown and light-gray; dark-gray, light-blue, light-green, light-cyan, light-cyan, light-red, light-magenta, yellow and white. Among these colors, only the first eight are used for the background. The optional highlight color pair, if specified, will be used to show the current menu selection. When not specified, GRUB will use the inverse of the normal colors.

The rest of the configuration file consists of the boot stanzas for our demonstration system. The title command marks the beginning of a new boot stanza, and its argument is the label that will be displayed for its entry in the menu, from the first non-white-space character to the end of the line. The remaining commands in each stanza are identical to those used when working from the GRUB command line. The exception here is that we no longer need to give a boot command; GRUB does this job without asking.

This example configuration gives only a sample of the many flexible uses of the GRUB boot loader. Besides multiple OSes, you can use GRUB to set up menu selections for test kernels, rescue kernels, different kernel options and so on.

All in all, the GRUB configuration file will be very similar to your `/etc/lilo.conf`. And after working with the GRUB command line and these examples, it should be a simple matter of firing up your favorite text editor and creating a menu configuration file suitable for your own system and preferences. Don't worry if it's not perfect the first time; you will see that you can make changes interactively, and the GRUB command line is always available as a fallback.

Once you've got your configuration file, mount your GRUB floppy again, and copy the file (say it has been saved as `mygrub.conf`) into the magic location:

```
cp mygrub.conf /floppy/boot/grub/menu.lst
```

Now when you boot with your GRUB floppy—presto!—you will be greeted with a lovely boot menu like the one in Figure 2. If you like, just stare at it for the few seconds it needs to count down from the timeout setting, and then it will automatically boot into your default OS. Or, use the arrow keys to highlight the OS you want to load and press return. Or, type **c** to go to the now-familiar GRUB command prompt. From the command prompt, press ESC to go back to the boot menu again.

It is also possible to edit the entries displayed in the menu. Typing **e** will open a simple vi-like editor interface for the highlighted entry. This allows you to adjust or add any settings to the configuration before booting. Any changes made here will then remain in effect for the duration of the GRUB session. To make permanent changes, you will later need to edit the configuration file on the boot disk, saving the changes with your text editor.

Play with your GRUB floppy configuration until you have it set up the way you like. After just a few system boots, you'll be slinging through GRUB like hashbrowns in a diner.

Hard Disk Installation

By this time you may be thinking, “Okay, GRUB has got it goin' on. But do I have to keep booting from this lame floppy all the time?” Of course not. Booting from floppy is for weenies.

The operation for installing GRUB on the master boot record of your hard disk is similar to the creation of a GRUB floppy. The one difference is that our floppy has all the resources GRUB needs in one place. That is, the boot image, support and configuration files are all on the floppy device MBR and /boot/grub/ directory. In a hard disk setup, you can choose where you want these resources to be.

For example, you could set up a /boot/grub directory on the first partition of your first hard disk and copy all GRUB's files into it as we did in our floppy setup. In our demonstration system, this would be the Win98 partition, and you may choose to do it that way if you want. But you can also set up the /boot/grub directory up in any device/partition on your machine with a filesystem supported by GRUB. In practice it is usually best to install this support directory in your most frequently used and/or most stable partition; that is, one that you aren't reinstalling all the time.

For this example, we will use the Slackware partition since this stays pretty stable, and I tend to do more tracking and installations in the Debian system.

Once this decision is made, everything else is simple. First, boot into Slackware, create the `/boot/grub` directory and copy GRUB's files into it (these are all the files that the GRUB build installed in the `/usr/local/share/grub/i386-pc` directory). Make sure to put your handcrafted `menu.lst` configuration file in here, too.

Next, start GRUB, either with the `grub` executable you built in Linux or by rebooting with the GRUB floppy. If GRUB starts in menu mode, press `c` to go to command-line mode. Enter the following commands at the `grub` prompt:

```
grub> root (hd0,1)
grub> setup (hd0)
grub> quit
```

You're done. Your system is now fully GRUB'd, installed in the MBR of your hard disk. Type **reboot** as root (or take the floppy out and jab the keyboard with the old three-prong) and watch just how fast GRUB comes up now!

A few words of explanation about these installation commands. The first, `root (hd0,1)`, tells GRUB to mount this device, in this case the partition with the Slackware installation. All files will now be relative to this device, so the GRUB installer will know to look for its support files in the `/boot/grub` directory we created in the Slackware partition.

The second command, `setup (hd0)`, is a simplified front end to GRUB's `install` command. Note in particular that we specify the device as `(hd0)` and not `(hd0,0)`. Device `(hd0)` results in GRUB's installation to the master boot record, which is what we want. If we had instead used `(hd0,0)`, GRUB would be installed to the boot sector of the first partition, rather than the MBR. The difference is crucial; your technical writer makes mistakes like this so you don't have to. While each partition can have a boot sector, your hard disk will have only one master boot record the BIOS loads every time you start your machine. Unless you are doing some kind of funky boot-chaining, like using LILO to boot GRUB, you will usually want to install GRUB in the master boot record.

When GRUB installs itself on a device, it first copies a small piece of itself to the MBR, which it calls `stage1`. Then it follows `stage1` with just enough information about where to find the rest of GRUB. In our example, GRUB will put `stage1` in the MBR, followed by a blocklist that points to the Slackware partition. GRUB will then find the rest of what it needs (its `stage2` files) in the `/boot/grub` directory.

To check this setup, just edit the menu configuration file in Slackware's `/boot/grub/menu.lst` at any time. Any changes will be reflected in the next boot.

Error Recovery

If you should foul up the hard disk installation somehow or want to uninstall GRUB from your system, here's what you need to know.

First, if you ever want to clean your MBR from whatever is installed there, the canonical method is to use the **fdisk** program from an MS-DOS boot floppy:

```
A:> FDISK /MBR
```

Of course, this isn't necessary if you just want to go back to LILO as your system's boot manager. In that case, simply make sure your `/etc/lilo.conf` file has a line that reads **boot=/dev/hda**. Then, when the rest of the `lilo.conf` file is the way you want, just rerun LILO. This will put LILO back on the MBR of your system.

If you install GRUB in the boot sector of a partition, instead of the MBR (such as specifying `setup (hd0,0)` instead of `setup (hd0)`), you may need to reinstall that OS's boot loader. In the case of DOS/Windows, this means running the **sys** command from your DOS/Windows boot floppy: **A:> SYS C:**.

If, this is a Linux partition, it is again effective to rerun LILO, where `/etc/lilo.conf` has a line in the boot stanza that reads **root=/dev/hda1**.

In general, most OSes will have a way to reinstall their partition's boot sector without doing a full reinstallation from scratch. (For FreeBSD, see `boot0cfg(8)`; for OpenBSD, see `installboot(8)`.)

In practice, especially if you followed through on the GRUB floppy examples, you should find that GRUB itself is one of the best rescue and system recovery tools in your toolkit. For example, if you have ever made a screwup in your `lilo.conf` file, you know you can be in for some major pain if your system won't boot. With GRUB, you always have a miniature, self-contained operating system that can recognize and mount different filesystems, find files, locate kernels and boot them, bringing your system back to life so you can work on it. At times like these, GRUB can save your bacon.

Conclusions

As is typical of GNU software, GRUB is rich with capabilities beyond what are described here. Some of these include:

- Remapping disks and partition hiding, so you can even run multiple versions of DOS/Windows, on other than the first hard disk.

- Network booting with BOOTP and DHCP protocols, to support multiboot schemes across a network and diskless operation.
- Keyboard remapping, disk geometry access, memory reading, I/O port and processor probes, password protection, decompression support, etc.

See the GNU information manual for more information on these topics. GRUB is under active development, and even more features are planned for future releases.

In this brave GNU world, with vast acreage of cheap hard disk and a glut of great free OSes available, you really need a flexible and user-friendly boot loader to manage them all. Grab GRUB and give it a go.

Why GRUB?



Wayne Marshall (guinix@yahoo.com) is a UNIX programmer and technical consultant currently living in Guinea, West Africa. When not grubbing about with computers, he enjoys taking the pirogue for day trips to the local islands near Conakry with his wife, Paula.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

An Introduction to DNS and DNS Tools

Neil Anuskiewicz

Issue #85, May 2001

The explosive growth of the Internet was made possible, in part, by DNS.

The domain name system (DNS) hums along behind the scenes and, as with running water, we largely take it for granted. That this system just works is a testament to the hackers who designed and developed DNS and the open-source package called Bind, thereby introducing a scalable Internet to the world. Before DNS and Bind, `/etc/hosts` was the only way to translate IP addresses to human-friendly hostnames and vice versa.

This article will introduce the concepts of DNS and three commands with which you can examine DNS information: **host**, **dig** and **nslookup**.

The DNS is a distributed, hierarchical database where authority flows from the top (or root) of the hierarchy downward. When *Linux Journal* registered `linuxjournal.com`, they got permission from an entity that had authority at the root or top level. The Internet Corporation for Assigned Names and Numbers (ICANN) and a domain name registrar, transferred authority for `linuxjournal.com` to *Linux Journal*, which now has the authority to create subdomains such as `embedded.linuxjournal.com`, without the involvement of ICANN and a domain name registrar.

When trying to understand the structure of the DNS, think of an inverted tree—the very structure of the UNIX filesystem. Each branch of the tree is within a *zone* of authority; more than one branch of this tree can be within a single zone. *Linux Journal* could choose to retain authority for `embedded.linuxjournal.com`, or they could delegate it down the tree to someone else who could make subdomains such as `zeus.embedded.linuxjournal.com`.

The software (usually Bind) that stores domain name information is called a domain *name server*. A single name server can be authoritative for multiple zones. All zones have a primary master and a secondary master name server that provides authoritative responses for their zones.

If you query a name server not authoritative for a particular zone, that name server will most likely return the correct information. This is because zone information propagates throughout the Internet, and name servers cache zone information for which they are not authoritative.

When you register a new domain name, transfer your old one to a new host or just make changes to the zone database file, it often takes several days for the new information to propagate completely. During that interim period, nonauthoritative name servers often temporarily cache stale information about your domain name.

You may wonder how you fit into this process when you use the Internet. Well, whenever you use the Web, Telnet, FTP, etc., your software uses the resolver (the client side of the DNS), which is a set of library routines compiled into programs such as Mozilla. When you type **www.linuxjournal.com**, the resolver sets up the query to the name server that does the work of translating **www.linuxjournal.com** to 207.178.22.49 so you can get to the web site.

DNS Commands

For comprehensive coverage of DNS and DNS commands, read the man pages and get one of the excellent DNS books on the market, such as O'Reilly's *DNS and Bind* and Sybex's *Linux DNS Server Administration*.

Zone file database records divide DNS information into three primary types: NS (name server) records, MX (mail exchange) records and A (Address) records. NS records indicate the name servers. MX records indicate the hosts that handle e-mail delivery; the priority (pri) number indicates the order in which mail servers are used, with the lowest number receiving the highest priority. The A (Address) records map hostnames to IP addresses, the real names of machines.

host

This is the simplest of the DNS commands. It is a quick way to determine the IP address of a hostname:

```
host www.linuxjournal.com
www.linuxjournal.com has address 207.178.22.49
www.linuxjournal.com mail is handled (pri=80)
by www.ssc.com
www.linuxjournal.com mail is handled (pri=10)
by mail.ssc.com
```

```
www.linuxjournal.com mail is handled (pri=40)
by cascadia.a42.com
```

The `-a` option will return all of the DNS information in verbose format, as seen in Listing 1.

Listing 1. DNS Information in Verbose Format with `-a` Option

Now that you know the IP address for `www.linuxjournal.com`, you might want to make sure the reverse lookup works. The reverse lookup checks to see if the reverse zone file maps the IP address to the hostname:

```
host 207.178.22.49 49.22.178.207.IN-ADDR.ARPA
domain name pointer www.linuxjournal.com
```

dig (domain information groper)

This powerful command gathers and returns DNS information in a format the name server can use directly. For this reason, `dig` is particularly useful in scripts. You will find it easy to query specific name servers with `dig`, making it a useful tool for narrowing down the source of DNS problems.

Suppose you have just transferred your domain name hosting from `old-host.com` to `new-host.com`. A customer sends you an e-mail saying he cannot reach your web site when he is logged into his ISP. You suspect the zone information simply has not had time to propagate. So, you find out what the NS records are for the ISP in question:

```
dig ns isp-in-question.com

;; ANSWER SECTION:
isp-in-question.com. 10H IN NS ns1.hugeupstream.com.
isp-in-question.com. 10H IN NS isp-in-question.com.
isp-in-question.com. 10H IN NS ns.isp-in-question.com.
isp-in-question.com. 10H IN NS ns.goodnameserver.com.
```

Then you check your company's web site against the ISP's name servers:

```
dig www.yourcompany.com @ns.isp-in-question.com

;; ANSWER SECTION:
www.yourcompany.com. 59m53s IN A 192.168.5.10
```

Wait a minute, that is your old IP address. It appears the DNS information has not fully propagated yet.

Next, you decide to see if `old-host.com` has removed the old zone information from their name servers. The “any” option will retrieve all the DNS information:

```
dig any www.yourcompany.com @ns.old-host.com

;; ANSWER SECTION:
www.yourcompany.com. 1H IN A 192.168.200.250
```

```
;; AUTHORITY SECTION:
yourcompany.com.      1H IN NS    webns.new-isp.com.
yourcompany.com.      1H IN NS    srvns.new-isp.com.
```

In this case the A record shows your new IP address for your web server, and it shows the new authoritative name servers for your domain name. This is the information you hoped to find.

These are the most useful dig query types: dig any (gathers all DNS information), dig ns (gathers name server information), dig mx (gathers mail exchanger information) and dig a (gathers network address information).

The dig command can also do reverse lookups with output formatted for the zone file:

```
dig -x 192.168.200.250
;; ANSWER SECTION: 250.200.168.192.in-addr.arpa.
4h11s IN PTR    www.yourcompany.com.
```

nslookup

You can use this tool as a single line command, or you can use it interactively, which distinguishes it from the other DNS commands. Once you have started nslookup, type **set all** to list the default options. As with dig you can choose the server (name server) you want to query, and you can decide the type of DNS information on which to focus.

Listing 2. Output with nslookup

Just as you can issue commands to nslookup interactively, you can also change the initial defaults by starting a .nslookuprc file. The format of the .nslookup is one command per line:

```
set type=NS
set domain=srvns.new-host.com
set timeout=10
```

Conclusion

The ARPANET (the precursor to the Internet) had a few hundred hosts throughout the 1970s. A single flat file called HOSTS.TXT contained all of the information for every host. System administrators periodically downloaded the file and placed the information into their /etc/hosts file; take a look at your own /etc/hosts to see roughly what that file looked like. However, this system was not scalable. The advent of DNS made the exponential growth of the Internet possible.



Neil Anuskiewicz lives and works in Portland, Oregon. When he is not sitting in front of a computer he likes to hike and climb. Write Neil an e-mail at neil@pacifier.com.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Linux Teleconferencing: Improving the Wireless Network

Izzet Agoren

Issue #85, May 2001

Izzet gives insight as to how industry juggernauts tackle the challenge of wireless teleconferencing.

A stated goal of the Third Generation cellular standards bodies (3G) is for the wireless network to function as a seamless extension to the Internet and other IP-based packet network services. The brunt of the ground-breaking work we still await has been passed from companies to standards bodies to the Internet Engineering Task Force (IETF). IP transparency extends a plethora of services to the mobile devices already available to those with a wire line to the Internet, as well as simplicity and ease of adding new and more creative services explicitly designed to enhance wireless experience. An obvious result is an increase in the demand for wireless minutes by customers of the 3G carrier members.

The bandwidth available for broadband data transmission is both restricted and costly. Licenses for the frequency spectrum needed to support 3G services have been auctioned to network operators in the UK for \$36 billion. More recently the Federal Communications Commission (FCC) raised a net \$17 billion in a similar auction to US-based operators. This prompts carriers to improve the efficiency with which they employ the spectrum. The use of IP as a transport mechanism for voice (e.g., VoIP) requires the wireless network to carry real-time multimedia, even as it struggles to do non-real-time multimedia. For example, a full rate voice encoder like the G721.1 used by GSM-type networks generates 30ms (milliseconds) of payload, straddled with 40 bytes of a combined IP, UDP and real-time protocol (RTP) header as prescribed in the H.323 International Telecommunications Union (ITU) protocol for the delivery of multimedia. The 30ms of speech payload typically translates to 20 bytes, thereby giving us an efficiency of only 33%.

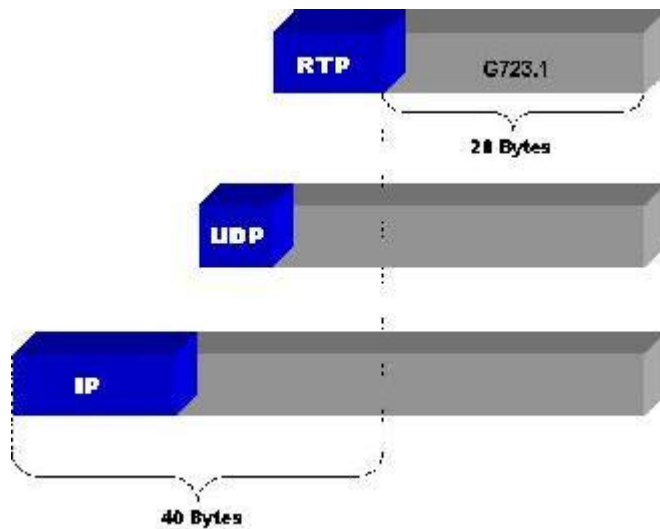


Figure 1. Spectral Efficiency of a H.323 Packet

Header compression algorithms proposed to the IETF reduce the header down to one byte in steady state but have varying degrees of hobbling complexity. The header compression algorithm is required to present both bit-exact payload and an encapsulating IP protocol header to the mobile device. This is referred to as loss-less compression.

The benefits presented by header compression prove to be an imperative measure in increasing the spectral efficiency of multimedia communications, particularly since CRTP, proposed in the late 1990s by Cisco, is deemed insufficient, or not robust enough, by the involved steering committees of the IETF, as well as the Third Generation Partnership Project (3GPP).

The details of which header compression algorithm the IETF will choose have not been fully determined. Nonetheless, the requirements have been hammered out and are summarized in Table 1. In fact, the current proposals meet many of the requirements but are peppered with intellectual property rights—something that defeats the objective of the IETF. A second round of proposals are under consideration so that different methodologies may be solicited. The current algorithms exploit the nature of the IP/UDP/RTP streams. Tables 3, 4 and 5 classify the nature of the header field for each protocol IP, UDP and RTP respectively. Table 2 defines the classifications.

Table 1. RoHC Requirements

Table 2. Header Field Classifications

Table 3. IP Header Fields and Classifications

Table 4. UDP Header Fields and Classifications

Table 5. RTP Header Fields and Classifications

The Essence of Compression Decompression

The essences of these algorithms are the strategies borne from these classifications. They are “never send”, “communicate at least once”, “communicate at least once or update”, “communicate update and/or refresh frequently”, “guarantee continuous robustness”, “communicate as is in all packets and establish” and “be prepared to update delta”.

Telecommunications companies that have pursued this approach to compress packet headers range from Nokia, Matsushita and Cisco to, most notably, Ericsson, for their heavyweight effort. The full details of their proposed algorithm have been submitted in IETF draft form and can be found at <http://www.dmn.tzi.org.org/ietf/rohc/>.

The Solution

The effects of the wireless channel and the response of the compression algorithms can be modeled using a small collection of hardware and software, namely a Linux box, a pair of H.323 generators and some freely available libraries. The rather simple setup required to simulate all of this can be done with just three computers, a pair of speakers and microphones, a few extra Ethernet cards and two cross cables. The software is just as minimal, and when used in conjunction with the setup shown in Figure 2, we obtain a particularly easily reproducible system.

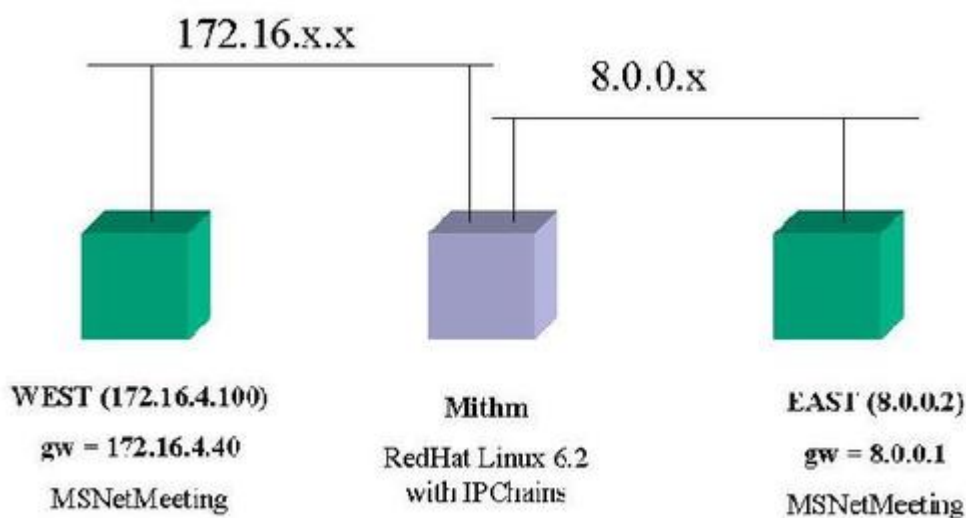


Figure 2. The Machine in the Middle Model

The open H.323 project is an excellent source of software components required for the generation of packets that comply with the standard. Two alternatives are available at the web site (<http://www.openh323.org/>) for both Linux and Windows platforms. For those of you who have Windows already, NetMeeting,

commonly bundled with the operating system, offers yet another version of an H.323-compliant multimedia engine.

Important scenarios need factoring into an implementation in terms of simulating the channel, including:

- Startup—establishing the session requires a great deal of robustness so that the compressor/decompressor pair can get into context as described in the previously mentioned strategies.
- Handoff—handoffs are a function of how fast the mobile device is moving, and the number of packets dropped follow a Poisson distribution with parameter nine. This behavior is considered graceful in that third-generation networks allow the session to survive the handover of the session without having to restart.
- Deep fade—these are the enemies of wireless communications and are caused by the hostile nature of spitting bits out on radio waves. Deep fades are typically attributed to the momentary shadowing of the radio signal as well the detrimental effect of interference experienced from congested areas. These are currently the major limiting factors under the academic microscope to the operation of all third-generation cellular networks.

Having installed the Linux machine in the middle of the network, we delve into the TCP/IP stack so that we can create the adaption functions in a modular, nonblocking, input/output, real-time, client/server fashion. In effect, we stop the packet flow by setting up rules in IP chains and using the libpcap interface to bring the stopped packets up to user space for analysis and, ultimately, compression/decompression. Having physically reproduced the architecture of Figure 2, it is a trivial matter to establish an uninterrupted teleconference session whose packets are forced through your Linux machine. IP forwarding must be enabled for this to function, particularly since we are going to stop the flow of packets with the use of IP chains. To verify that IP forwarding is indeed enabled, type:

```
echo /proc/sys/net/ipv4/ip_forwarding
```

The result is equal to one if we are ready to forward packets. The next step is to verify we can stop and restart the stream without killing the session. This means TCP packets must be uninterrupted. At this point, we need only concentrate on IP/UDP/RTP packets. The following commands will stop and restart the stream:

```
ipchains -P forward -DENY  
ipchains -P forward -ACCEPT
```

The -P option is indiscriminate of protocol; it will stop ICMP, ARP, TCP and UDP packets.

Now that we can play with the stream we can be selective with which ones we transmit and how we transmit them.

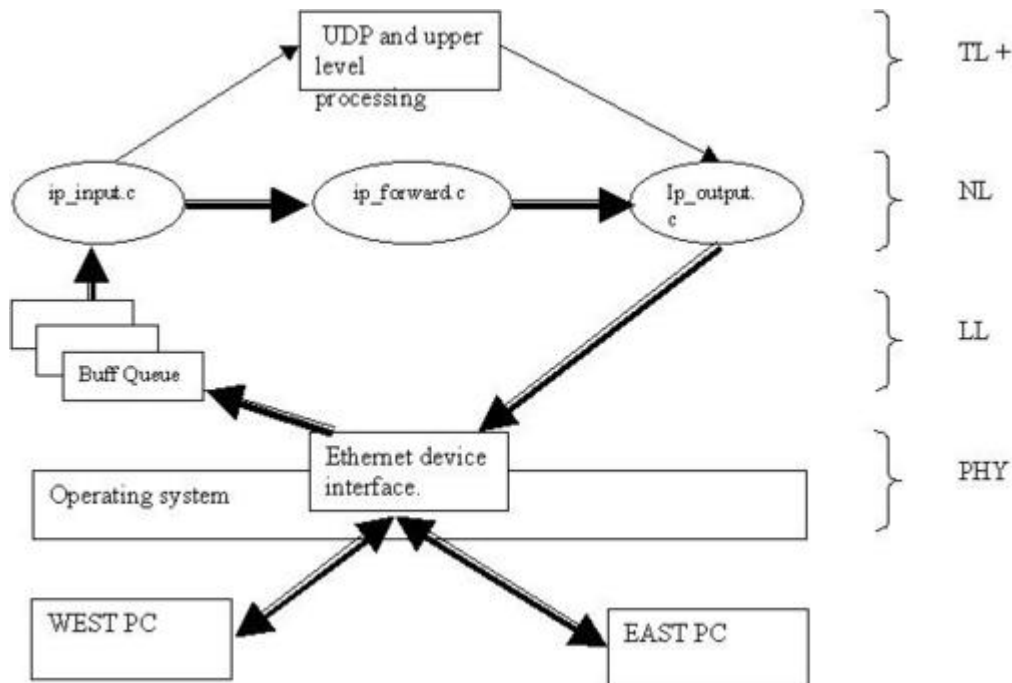


Figure 3. The Linux TCP/IP Stack

The link layer (LL) is where we want to pick up our packets in order to retain all the IP fields. The packets are received by the networking stack and queued in a linked list structure called sk_buff where they are serviced automatically by the top-half software interrupts of the kernel in ip_input.c, ip_forward.c and ip_output.c. For a more in-depth treatment of how socket buffers are managed in memory, see Alan Cox's "Network Buffers and Memory Management" (*Linux Journal*, October 1996). Most user-space programs interface the networking stack via Berkeley Packet Filter (BPF) or INET sockets. For security purposes, these socket interfaces were not designed to delve down to the Ethernet or device/physical layer (PL). A compromise is reached by opening a raw socket that retains the IP fields by interacting directly with the IP layer of the stack. Although reading packets in their raw form is supported by Libpcap, the ability to transmit them is only feasible through modifications to Libpcap itself. The definitive text on TCP/IP networking stack can be found in *UNIX Network Programming, Vol.1*, by W. Richard Stevens. For a more Linux-specific treatment of the subject, the interested reader is referred to David A. Rusling's "Chapter 10, Networks" in *The Linux Kernel*, which can be found at www.linuxhq.com/guides/TLK/net/net.html.

A succinct way of circumventing the necessary changes to either the kernel or libpcap in order to pop the raw packets in full, to and from the Ethernet device, is found in the form of a Perl5 CPAN package, namely RawIP (<http://quake.skif.net/RawIP/>). Figure 3 is a diagrammatic representation that maps the Linux TCP/IP stack and the code in the kernel (2.2.x) responsible for dealing with the packet flow to be compressed.

Listing 1 [at [L's ftp site](#)] is a rudimentary means, using Perl5, of picking up the stopped packets, outputting the IP id field contents and, in turn, passing them on to their final destination. The IP address of source and destinations are the only parameters required. The script is discriminate of the protocol, so we are now able to concentrate on just the voice or even, as it turns out, video packets.

The script creates a text file id_dump.txt that gives a handle on the IP id during any given session. Extending this to the other fields by quoting them in the script is the first stage needed in creating a state machine that implements any one of the proposals submitted to the IETF working group. Listing 2 uses the Math::Random Perl5 CPAN package to introduce an average packet or, in the case of VoIP, frame error rate according to a uniform distribution of 20%. The effects of which are immediately distinguishable when used in conjunction with the gateway of Listing 1, which will now begin to take the form of a high-level wireless channel simulator. The justification of deeming this a sufficient means of corrupting the stream is two-fold. A very precise 3G wideband code-division multiple access modulating channel model with multiple Raleigh fading paths is freely available for download from w3.antd.nist.gov/wctg/3G/3G.html. Its use, while it comes highly recommended, can have a tremendous impact on the real-time nature of the session in the absence of a Beowulf cluster. Furthermore, the delay will reduce the subjective nature of system performance that engineers often rely upon in the case of the perception of voice transmission quality.

Listing 2. Perl5 Frame Error Rate Thanks to Knuth

Now that you are armed with an all applicable framework, the next step would be to apply the work to the analysis of video packets and their resilience to our hostile channel or pack this into an embedded system for the creation of a low-cost teleconferencing tool.



Izzet Agoren is an Electrical Engineering Master's candidate at the Center for Information and Communications Technology Research, Pennsylvania State University. His focus is the study of using the forthcoming third-generation mobile networking technologies for the delivery of real-time multimedia services.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

JavaServer Pages

Reuven M. Lerner

Issue #85, May 2001

This month Reuven discusses JSPs, which allow programmers and nonprogrammers alike to create servlets and dynamic pages with a mix of Java and HTML.

Last month, we took a first look at server-side Java, sticking our toes into the water by writing some servlets. Servlets are Java programs that produce dynamic web content. CGI programs are executable programs external to the web server that execute from scratch each time they are invoked. By contrast, Java servlets live inside a servlet container, a Java virtual machine (JVM), that is closely connected to an HTTP server. Whenever the web server needs dynamic content, it makes a request from the servlet container.

In many ways, writing a Java servlet is like writing a mod_perl handler: it gives you a great deal of power, but also requires a fair amount of discipline. It can also be frustrating to write a servlet that is 90% static HTML and 10% Java, and the number of times that you invoke `out.println()` can become maddening.

An increasingly popular solution to this problem is JavaServer Pages or JSP. JSP is similar in spirit to Microsoft's ASP, as well as to the open-source PHP language and Mason component system. JSP allows you to mix Java with HTML in a number of different ways. The fact that JSPs, like most Java programs, are remarkably platform-independent, means you can write JSPs on a Windows box, run them on a development Linux server and then deploy them on Solaris.

This month, we will take a quick look at JSPs, which are a good way to get to learn Java as well as an easy way for Java programmers to create servlets without having to work too hard.

How JSPs Work

The idea behind JSPs is remarkably simple: they are servlets in disguise. When a JSP is first invoked, it is automatically turned into a servlet. This servlet is then compiled into a Java .class file, which is then executed inside of the servlet container. The first time a JSP is invoked, it will take a little bit longer to return data to the user, due to all the action taking place behind the scenes.

In a JSP, everything is assumed to be static content unless it is placed inside special braces, `<% and %>`. These tags are known as “scriptlets” in JSP lingo. The following HTML file (which we will name `main.jsp`) is also a perfectly legitimate JSP:

```
<HTML>
  <Head>
    <Title>Static JSP Title</Title>
  </Head>
  <Body>
    <P>Static JSP Content</P>
  </Body>
</HTML>
```

The above JSP is rather boring in that it consists exclusively of static content. But the JSP engine doesn't care how many pieces of dynamic content a JSP contains; it will turn the entire thing into a servlet regardless of its complexity. In the case of the above JSP, the resulting servlet will be little more than a long string of `out.println()` statements inside of the `doGet()` method.

On my system, I saved the above HTML into `main.jsp` in the `examples` directory that comes with Tomcat `/usr/java/jakarta-tomcat-3.2.1/webapps/examples/jsp/`. This is admittedly not the best place to install it, but it is the easiest.

Once I've installed the JSP, I don't need to do anything else; the system will automatically translate it into a servlet source (.java) file, and then compile it into a Java .class file.

We can execute and view our JSP via the Tomcat server, which operates by default on port 8080, `http://localhost:8080/examples/jsp/main.jsp/`.

If you've configured Apache and `mod_jk` to forward servlet and JSP queries to Tomcat, then you should also be able to view `main.jsp` with this URL: `http://localhost/examples/jsp/main.jsp/`.

On my system, the .java and .class files generated by the JSP system for `main.jsp` are in the directory: `/usr/java/jakarta-tomcat-3.2.1/work/localhost_8080%2Fexamples`. If I list the contents of this directory, I see the following:

```
_0002fjsp_0002fmain_0002ejspmain.class  
_0002fjsp_0002fmain_0002ejspmain_jsp_0.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_1.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_2.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_3.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_4.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_5.java  
_0002fjsp_0002fmain_0002ejspmain_jsp_6.java
```

As you can see, there are seven different .java files, each corresponding to a different version of the original JSP. Each time I modify the JSP, the system must create a new .java file. The Tomcat default keeps previous versions of the JSP-based servlet around; however, there can only be one .class file at a given time, which is clearly the case in this directory.

The names of the .java and .class files are quite long and aren't meant to be entered directly into a web browser. Part of the magic of JSPs is that Tomcat can find the servlet associated with a given URL intelligently and automatically, creating the Java source file as necessary.

You should take a look at the Java source code created by the JSP translator so that you can get a feel for the hard work being done behind the scenes. Our simple, static JSP has been turned into a servlet that takes up more than 100 lines of Java source code. In case we ever have to debug our JSPs from the translated servlet source code—a difficult task, as anyone who has used Perl's HTML::Mason can attest—the servlet includes comments that provide a basic mapping from the line numbers in the original JSP to those in the resulting servlet.

Dynamic Content

We can jazz things up a bit by adding one of the special JSP tags to main.jsp. The first tag inserts the results of some Java code into the output sent to the user:

```
<HTML>  
  <Head>  
    <Title>Mini-dynamic JSP Title</Title>  
  </Head>  
  <Body>  
    <P>You are connecting from  
      <%= request.getRemoteHost() %>.</P>  
  </Body>  
</HTML>
```

The expression inside of `<%= %>` is invoked, and its return value is placed in the output stream. Because a JSP is a servlet in disguise, it has access to the objects normally available to a servlet, such as “request” and “response”. Notice how the Java within `<%= %>` does not end with a semicolon; I can tell you from personal experience that it's difficult to break the habit of inserting semicolons there, but your JSPs will die if you insist on them.

To perform one or more Java computations without having the results sent to the user's browser, use the basic `<% %>` tags. These tags can be interspersed with HTML, making it possible to have conditional text appear in the response (see Listing 1).

Listing 1. Using `<% %>` Tags

If the hostname of the user's computer is available, we print its name. Otherwise, we print the host's IP address. Notice how the if/then/else block is interspersed with the static HTML. The `request.getRemoteAddr()` call is invoked only if `request.getRemoteHost()` returns an empty string (`""`).

A number of JSP directives are invoked with the `<%@ %>` tags. All of these directives take effect at the time of JSP-to-servlet translation. A directive keyword is placed immediately after the `@` symbol, followed by zero or more attributes.

For example, let's assume we have a standard sitewide menu bar in a JSP named `menubar.jsp`:

```
<table>
<tr>
  <td><a href="one">Option 1</a></td>
</tr>
<tr>
  <td><a href="two">Option 2</a></td>
</tr>
</table>
```

We can incorporate that into our document using the "include" directive (see Listing 2).

Listing 2. Using the "Include" Directive

It's important to remember that directives take effect when the JSP is turned into a servlet, not at runtime. Thus, the above example will work fine until you change `menubar.jsp`. Since the contents of `menubar.jsp` were incorporated into `main.jsp` just before the latter was turned into a servlet, the `<%@ include %>` tag no longer exists and, therefore, will not update things in the way we might expect. The solution is to use a runtime JSP action, described below.

There are two additional special JSP tags. One of them, `<%-- --%>`, acts as a comment. While it might seem odd to use a JSP comment when HTML comments already exist, the difference is important to remember: JSP comments are removed by the JSP engine when the servlet is created. By contrast, HTML comments are passed through untouched and are visible to any end user who selects the "view source" option on their browser. Given the choice, I tend to put most comments inside of the JSP comment tags, except for

those that will help me to debug the JSP's output using the resulting HTML source code.

The final JSP tag, `<%! %>`, allows you to declare instance variables (also known as fields) for the JSP's resulting servlet. While it might seem tempting to use declaration tags to declare variables you will use in the rest of the JSP, remember that using fields means you must deal with thread safety. Given the headaches associated with threads, it's probably a good idea to avoid them if you can. You can also use the declaration tag to define new methods local to your JSP, although I'm not convinced this is such a good idea.

JSP Actions

Directives are useful if we want to affect the way in which the servlet is built. But what if we want to affect the servlet's actions at runtime?

We could, of course, include Java code to perform these actions. But JSP includes a number of special tags that are translated into Java code in the servlet, allowing you to write code without having to know any Java.

JSP action tags are actually XML and are defined in special XML documents known as tag libraries. So while they might appear to be HTML, they aren't, which often means you must pay particular attention to items such as closing tags and slashes.

The built-in JSP tag library includes a number of functions, one of which looks suspiciously similar to the include directive we saw in Listing 2. Listing 3 shows a version of main.html that uses the `<jsp:include>` action, rather than the directive, to bring in menubar.jsp.

Listing 3. Using the `<jsp:include>` Action

The difference between this version and its predecessor is subtle but significant: whereas the include directive incorporates the named page when the JSP is translated into a servlet, the include action works at runtime. If you were to modify menubar.jsp between invocations of main.jsp, the directive version would ignore the new menu bar, while the action version would display the latest version. Of course, this comes at a cost; the `<jsp:include>` action performs a runtime request, making it slower and less efficient than the directive.

Because pages requested by `<jsp:include>` have access to all of the request information from the top-level JSP, it's possible to use `<jsp:include>` to create dynamically changing menu bars, personalization systems and database access libraries.

There are other JSP actions as well. One of them is `<jsp:forward>`, which passes the request onto another JSP. As with `<jsp:include>`, this takes place within the servlet engine, meaning HTTP request and user-session information are still available. For example, Listing 4 shows a version of our JSP that forwards users to another page if their hostname is not identifiable.

Listing 4. Sample Version of JSP

If your server does not contain a page named `no-reverse.jsp`, then the user will get a 404 (file not found) error in their web browser. However, their browser will continue to display the URL of the originally requested page, `main.jsp`. This is because the JSP forward is performed internally, without the need for an external HTTP forward.

A Simple Web Log JSP

Last month, we wrote some simple servlets that allow us to create and view a web log, sometimes known as blog. Since JSPs are translated into servlets, there is no reason why we cannot create a JSP that accomplishes the same thing as that servlet. It will obviously look a bit different, but the effect should be the same.

Listing 5 contains a JSP (`showblog.jsp`) that performs the same task as the `ShowBlog` servlet from last month. In other words, this JSP prints the contents of my web log, as stored in a PostgreSQL database table, sorted from the newest entry to the oldest.

Listing 5. `showblog.jsp`

I should note right now that `showblog.jsp` is a terrible example of how to write JSPs; it is simply meant to demonstrate what is possible, not what is elegant or best. (We'll discuss such issues over the next two months, when we discuss JavaBeans and custom tag libraries.)

Let's go through this JSP, so that you can see exactly how it works.

We begin with two "page" directives. These directives allow us to set up the basic configuration for our JSP, beginning with the MIME `ContentType` header the page will return, and even permit us to specify the programming language to be interspersed with the nonprogrammatic text. While such functionality does not actually exist, it's theoretically possible to write JSPs that use Perl to produce XML, or Python to produce PNG images.

Notice how we can name one or more attributes in our page directive. The first line of `showblog.jsp` sets both the language and the `ContentType` attributes.

The second line indicates that the resulting servlet should import the packages within `java.sql`, which will allow us to connect to our relational database server (PostgreSQL in this case) using JDBC.

After a tiny bit of introductory HTML, we drop deep into Java. We create an SQL connection object and use it to connect to our PostgreSQL server. We retrieve data from the database, and then iterate over the `ResultSet` on a row-by-row basis.

Something Is Wrong Here

As I indicated above, this is a terrible way to write JSPs. Not only will the performance be terrible because of the many database connections being created and destroyed, but we have created a horrible mishmash of code and HTML. Indeed, the only thing we seem to have saved here is a bit of effort writing `out.println()` for producing HTML output.

Moreover, much of the intent behind JSPs is to remove code from the HTML pages, allowing nonprogrammers to create dynamic pages with a minimum of effort. If we must insert this much code in order to create dynamic pages, chances are fairly slim that a nonprogrammer will want to try their hand at web development.

Moreover, our translation of the `ShowBlog` servlet to a JSP resulted in the removal of several exception-handling routines. Our servlet was intelligent enough to handle the disappearance of the PostgreSQL server and could produce a reasonable error message. Our JSP, by contrast, produced a backtrace containing error messages. This backtrace is useful for developers but is neither friendly nor useful for the end user who will visit our site (in all fairness, we can set the `errorPage` attribute in the page directive, such that errors are forwarded to a different JSP).

A good solution would remove as much code as possible from JSPs, allowing nonprogrammers to make use of that code in a standard way and separating the programmatic and nonprogrammatic content. And indeed, JSPs come with support for `JavaBeans`, where each “bean” is actually an object with a variety of methods we can use from within a JSP, using special `<jsp: >` actions. The trick to a successful JSP deployment depends on, in no small part, an intelligent use of `JavaBeans`. Moreover, JSPs allow us to create our own tag libraries defining custom actions, so that we can replace even more code with tags that resemble the assorted `<jsp: >` actions.

Conclusion

JavaServer Pages, or JSPs, provide a coating of syntactic sugar around servlets, which can sometimes be difficult for nonprogrammers to learn. However, this month's most complex example of a JSP demonstrates that they can easily get out of control, containing close to as much code as a typical servlet. While it is easier to work with JSPs overall, the supposed benefits of separate code from HTML fall by the wayside as the JSPs get more complex.

Next month, we will look at JavaBeans, which allow us to push quite a bit of programming to classes defined and maintained outside of the JSP. Following that, we will look at JSP's custom tag libraries, which make it possible to create our own little languages for use inside of our JSPs.

Resources



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

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Fly Me to the Wine Cellar

Marcel Gagné

Issue #85, May 2001

Marcel demonstrates how Linux can get you literally off the ground with the FlightGear simulator.

Not bad, eh, François? What do you mean, what is it for? This month's issue of *Linux Journal* deals with training and certification, and I am exploring that very topic. What? Of course I am on topic. This is a flight simulator, is it not? No one said anything about what kind of training and certification I should be cooking up. *Mon Dieu!* You are a wonderful waiter, *mon ami*, but you can be so literal. Watch this barrel roll, François. Wonderful, *non?*

Quoi? Ah, our guests are here. Welcome, *mes amis*. Sit down and make yourselves comfortable. We were discussing flight training and certification with the help of your Linux system. Of course, you can stretch your virtual wings at your computer, but to really fly, you must visit your local flight school and enroll, as your humble chef has done. *Oui, mes amis*. A number of years ago, I made my way to the local airport and took the first steps toward getting my pilot's license. It is a wonderful experience and one I highly recommend. Most schools will offer an inexpensive familiarization flight to give you a feeling of what you can expect when you take to the skies.

François. What are you doing standing there? *Vite, François!* To the wine cellar. In honor of the international nature of this topic, I suggest you bring up the 1997 Chilean Petite Syrah for our guests. When the world is open to you, it is only fair to be open to the world, *non?*

While the necessary skills of the pilot must eventually be honed in the pilot's seat, you can use your Linux system to help you with the process. For instance, you can download and build software to help you with your flight planning. Getting from one airport to another is more than simply following the road. For one thing, there is no road.

Ah, François, you are back. Please pour for our friends. While you are enjoying your Syrah, you might want to look at the first item on the menu, and what an item.

Even licensed pilots need to keep their skills tuned. In the commercial aviation business, this is often done with flight simulators. As expensive as these things can be, it is certainly less expensive than taking the old 747 out for a spin, as they say, *non*? For the Linux user, there are a few simulators available. We are going to explore one in particular.

Probably the single most ambitious simulator project out there is *FlightGear*. The developers of this incredible package have produced a very impressive flight simulator. The scenery itself is breathtaking, and the coloration of land and sky verges on photo-realistic. *FlightGear* takes the notion of realism so seriously they have started the arduous process of applying for FAA certification and approval. This realism comes at a cost, however. Trying to run *FlightGear* with anything short of a decent accelerated 3-D video card will be a slow and painful experience. If you are lucky enough to have such a card, then you must experience *FlightGear*. Fly, don't run, to <http://www.flightgear.org/> to download your copy.

When I picked up my copy for this article, the release was 0.7.6. The installation process requires that you have a few libraries installed on your system, including PLIB, SimGear and Mesa. The GLUT libraries are also required, but they are part of the latest Mesa 3-D graphics library, so that should save you a step. Mesa is now distributed with a number of Linux CD distributions, so check yours out before going to look for the source. Links to all these sites are in the Resources section at the end of this article.

At the heart of this whole process is 3-D rendering, so we need to start with Mesa. If you do not have it on your system, go to <http://www.mesa3d.org/> and get a copy.

Once we have verified the presence of Mesa, we have to create the PLIB libraries. I found the install to be much better with the latest version (1.3.1). Start by extracting the source to a temporary build directory. I like to use `usr/local/src` for that purpose, hence the first command below:

```
cd /usr/local/src
tar -xzvf plib-1.3.1.tar.gz
cd plib-1.3.1
./configure
make
make install
```

Similarly, you should then extract and build the SimGear libraries. The version at the time of this writing was SimGear-0.0.14.tar.gz. We build it like this:

```
tar -xzvf SimGear-0.0.14.tar.gz
cd SimGear-0.0.14
./configure
make
make install
```

Next, unpack the *FlightGear* base package. The default location for the root directory is `/usr/local/lib`, and this is where I chose to unpack the archive. Note that this unpacking does not require you to do any compilation. These are base files for the packages, including some default aircraft models, scenery and so on:

```
cd /usr/local/lib
tar -tzvf fgfs-base-0.7.6.tar.gz
```

This will create a directory called `FlightGear`. Finally, we have the package itself. Extract your `FlightGear-0.7.6.tar.gz` file into a temporary directory and build the software:

```
tar -xzvf FlightGear-0.7.6.tar.gz
cd FlightGear-0.7.6
./configure
make
make install
```

To take advantage of hardware acceleration, the *FlightGear* executable (**fgfs**) needs to run SUID root:

```
chmod +s /usr/local/bin/fgfs
```

If all has gone well, you should be able to start your flight simulator from the comfort of your desk chair. I poured myself a glass of wine and fired up my new flight simulator. Yes, I know *mes amis*, one does not drink and fly, but this is a simulation. Perhaps I will pour a gingerale instead.

```
fgfs --fg-root=/some_directory/FlightGear
```

By the way, you need to specify the `--fg-root` flag only if you have chosen to install the *FlightGear* base files (scenery, etc.) somewhere other than its default location. For a look at my Cessna 172's cockpit, have a look at Figure 1.



Figure 1. Marcel Attempts a Takeoff from Toronto International Airport

I will let you in on a secret, *mes amis*. Your humble chef is also a licensed pilot, and he will admit right here that a good flight simulator can be quite intimidating when you are still trying to get the hang of it. Once you get comfortable with the controls (I used the keyboard to control my airplane), it is time to decide where to go. After all, it is a big planet, *non?*

Luckily, there is no lack of places to fly with *FlightGear*. Click on their download page and scroll to the Additional Scenery section. One click will transport you to a map of the world broken up into 10x10 degree chunks. In my case, I wanted the scenery that covered Toronto International Airport. That was the `w080n40.tar.gz` file. To be able to use that scenery, you need to change directory to the *FlightGear* scenery directory. Remember the default installation for the base stuff was in `/usr/local/lib`:

```
cd /usr/local/lib/FlightGear/Scenery
tar -xzvf w090n40.tar.gz
```

From there, I can restart my session from the Toronto airport by specifying the airport's identifier:

```
fgfs --airport-id=CYYZ
```

Unfortunately, we will not be able to learn everything there is to know about flying from this little introduction to *FlightGear*, so I do recommend you take some time and have fun with it.

If helicopters are more your passion, another great project under way is *Search and Rescue*. Not only do you get to fly a variety of helicopters and practice your flying, but you also take on search and rescue missions (hence the name),

saving victims from terrible fates. Ah, the pressure. Have a look at Figure 2 for a view of my Sikorsky Jayhawk on the pad at LA International.

The process of installing the software is essentially the same as our previous examples. Unpack the source from the SearchAndRescue0.7b.tgz bundle into your temporary directory. Note that you will also need the xviv libraries for this to compile properly. For sound and joystick control, you will also need a couple of additional libraries available off the web site, but these are not necessary for the program to work:

```
tar -xzvf SearchAndRescue0.7b.tgz
cd SearchAndRescue0.7b/sar
make -f Makefile.Linux
make install
```

The default installation directory for the binary is /usr/games. To start *Search and Rescue*, simply type the following (note the capitalization):

```
/usr/games/SearchAndRescue
```

While offering a very realistic view, I found that *Search and Rescue* was quite playable without video acceleration.



Figure 2. My Sikorsky Readies for Takeoff at LA International Airport

Once you have mastered all aspects of the sky, only the stars are beyond your reach. Or are they? It is true that we may not have access to the universe in the next week or so. François is still working out the bugs in the faster-than-light travel program he is developing. Relax François. It is just a little joke, *non*? Nevertheless, you can still experience the thrill of interstellar flight (and combat) with one of the single most addictive games on your Linux system—one you may not even be aware you have.

It is called *XPilot*, and if it is not already on your machine, check your distribution CD. If that still fails, head on over to <http://www.xpilot.org/> and download your own copy. You might also want to have a look at the *XPilot*

Newbie manual on that site. But before we go on I feel I must caution that, like wine, you should take *XPilot* in moderation.

That said, it is time to play. To find a list of servers that you can join, Telnet to this address:

```
telnet meta.xpilot.org 4401
```

Note the port number at the end (4401). You might want to direct this output to a file (**telnet meta.xpilot.org 4401 > xpilot.servers**) because there will be a lot of information scrolling past your screen. By doing a Telnet to port 4402, you can get the latest FAQ on the game. A Telnet to port 4400 puts you into interactive mode with the server. From there, you can type **HELP**, or **LIST** or even **FAQ**. Here's a sample from that server list:

```
4.2.1 :marach.dd.chalmers.se :15345 :3 :26d
4.2.1 :tigger.skaro.bt.co.uk :15345 :1 :0.03 :The Globe
```

The first part is the version number, followed by the hostname, port number, number of users, server uptime and map name. That's right, *mes amis*, these games are live, played in real time (so to speak) with others from around the world. Even better, those who might feel left out because they do not yet have that high-speed 3-D graphics card, do not worry. If you have X running on your system, you can play *XPilot* and pit your skills against the best star pilots the world has to offer. Let's pick "Tourmination" (since there are already three players active) and join that game. To do this, I type the command:

```
xpilot -name francois -join marach.dd.chalmers.se
```

The `-name` option is a nickname I pick for myself for the duration of the game (which lasts only moments because some slimy person blasts my fighter into space dust). As you may imagine, many options can be set. Type **xpilot -help** for a sample. Then, have a look at Figure 3 to see a picture of my short-lived battle in space.



Figure 3. Marcel's Ship Is Blasted to Space Dust in XPilot

Oui, François? Mon Dieu! Is it that time already? Before you go *mes amis*, let me give you one other little additional flying treat. You probably all know about the xscreensaver package with its fireworks displays, dancing lines, cool fractal effects and, most importantly, the warp journey through space dodging rocks as your screen rotates this way and that. The package is probably already installed on your system. If you do not have it, check your distribution CD-ROM or look in the Resources section. Good. Now, look around the house and dig up those old 3-D glasses. You know, the ones with one blue lens and one red lens. Now, put on the 3-D glasses, fasten your seat belts, and type the following command:

```
/usr/X11R6/lib/xscreensaver/rocks -3d
```

It is just like being there, *non?* You may not know this, but you can execute all these programs individually (as with our rocks) without having to wait for your screen saver to kick in. That is it for now, *mes amis*. François, please be so kind as to pour our friends a final glass of wine before they go. And you, *mes amis*, must admit that it is fun to have an operating system that really flies; now you can fly with it, too. A little joke, but not very good, *non? François, mon ami*, better fill my glass too.

Until next time, please join us here at *Chez Marcel*. Your table will be waiting.

A votre santé! Bon appétit!

Resources



Marcel Gagné lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy, and is coeditor of *TransVersions*, a science fiction, fantasy and horror anthology. He loves Linux and all flavors of UNIX and will even admit it in public. 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/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Checking Your Work with Scanners, Part I (of II): nmap

Mick Bauer

Issue #85, May 2001

Using nmap to test your system's potential weak points.

You may have heard horror stories about how easy it is for evil system crackers to probe potential victims' systems for vulnerabilities using software tools readily available on the Internet. The bad news is, these stories are generally true. The good news is, many of these tools are extremely useful (and even designed) for the legitimate purpose of scanning *your own* system for weaknesses.

This and next month's column will explore some of the ways ordinary system administrators and high-powered security professionals alike can use **nmap** and **nessus**, two outstanding open-source packages, to improve system security. But remember, knowledge is power, and it's up to you to use it responsibly (and in ways that won't compel any men in black to confiscate your beloved Red Hat boxen).

Why We (Good Guys) Scan

Why scan? If you're a system cracker, you scan to determine what services a system is running and to which well-known vulnerabilities those services appear to be subject. If you're a system administrator you scan for essentially the same reasons but in the interest of fixing (or at least understanding) your systems, not breaking into them.

It may sound odd for good guys to use the same tools as the bad guys they're trying to thwart. After all, we don't test our door locks by trying to kick in our own doors. But system security is exponentially more complicated than physical security. It's nowhere near as easy to gauge the relative security of a networked computer system as it is the door to your house. While ideally we should always know which network processes are active on every host we

administer, the fact is that in this age of hyper-connectedness it's difficult to keep track of this information.

Therefore, we security-conscious geeks are obliged to take seriously any tool that can provide some sort of sanity check, even an incomplete and imperfect one (as is anything that tries to measure a moving target like system security), despite or even because of that tool's usefulness to the bad guys. However, we also need to remember that neither a security scanner nor any other single tool or technique will magically grant total security. Repeat after me: security is a process, not a product.

Having said all that, there's one more reason we dig security scanners: notoriety. It's fun to pretend to be a s00p3r 3L33T HaX0r. Using tools like nmap and nessus while saying with a straight face that we're "working" is priceless!

Types of Scans and Their Uses

There are basically two types of system scans. Port scans look for open TCP and UDP ports, for "listening services". Security scans go a step further and probe identified services for known weaknesses. In terms of sophistication, doing a port scan is like counting how many doors and windows a house has; running a security scan is more like rattling all the doorknobs and checking the windows for alarm-sensors.

Oh, I almost forgot ping sweeps, arguably a third kind of scan. These are done to identify which IPs in a given range or network are active (i.e., which hosts respond to pinging). While this can be useful depending on the task at hand, in the interests of keeping this month's column to a manageable level, I'll focus on single-system port and security scanning. Everything we discuss here applies whether you're scanning five hosts or 65,500.

nmap, World Champion Port-Scanner

The basic premise of port scanning is simple: if you try to connect to a given port, you can determine whether that port is closed/inactive or whether an application (i.e., web server, FTP daemon, etc.) is accepting connections there. As it happens, it is easy to write a simple port-scanner that uses the local connect() system call to TCP connections on various ports; with the right modules, you can even do this with Perl. However, this method also happens to be the most obtrusive and obvious way to scan, and it tends to result in numerous log entries on one's target systems.

Enter nmap, by Fyodor. nmap can do simple connect() scans if you like, but its real thing is "stealth scanning". Stealth scanning involves the use of ersatz TCP

packets designed to trigger a response from each target system without actually completing a TCP connection.

nmap supports not one, but four different kinds of stealth scans in addition to TCP Connect scanning, UDP scanning, RPC scanning, ping sweeps and even operating system fingerprinting. It also boasts a number of features more useful to black-hat than white-hat hackers, such as FTP-bounce scanning, ACK scanning and Window firewall scanning, but those are of little interest to *Linux Journal's* highly ethical readers. In short, nmap is by far the most feature-rich and versatile port-scanner available today.

Here, then, is a summary of the most important types of scans nmap can do:

TCP Connect Scan—uses the OS's native `connect()` system call to attempt a full three-way TCP handshake (SYN, ACK-SYN, ACK) on each probed port. A failed connection (i.e., if the server replies to your SYN packet with an ACK-RST packet) indicates a closed port. It doesn't require root privileges and is one of the faster scanning methods. Not surprisingly, however, most server applications will log connections that are closed immediately after they're open, so this is a fairly "noisy" scan.

TCP SYN Scan—two-thirds of a TCP Connect scan; if the target returns an ACK-SYN packet, nmap immediately sends an RST packet rather than completing the handshake with an ACK packet. "Half-open" connections such as these are far less likely to be logged, so SYN scanning is much less perceptible than TCP Connect scanning. The trade-off is that since nmap, rather than the kernel, builds these packets, you must be root to run nmap in this mode.

TCP FIN Scan—rather than even pretending to initiate a standard TCP connection, nmap sends a single FIN (final) packet. If the target's TCP/IP stack is RFC-793-compliant (MS-anything, HP-UX, IRIX, MVS and Cisco IOS are not) then open ports will drop the packet and closed ports will send an RST.

TCP NULL Scan—similar to a FIN scan but instead a TCP-flagless packet (i.e., a null packet) is sent. Also relies on the RFC-793-compliant behavior described above.

TCP Xmas Tree Scan—similar to a FIN scan but instead sends a packet with its FIN, PSH and URG flags (final, push data and urgent, respectively) set. It also relies on the RFC-793-compliant behavior described above.

UDP Scan—because UDP is a connectionless protocol, i.e., there's no protocol-defined relationship between packets in either direction, UDP has no handshake to play with as in the TCP scans described above. However, most

OS' TCP/IP stacks will return an ICMP "Port Unreachable" packet if a UDP packet is sent to a closed UDP port. Thus, a port that doesn't return an ICMP packet can be assumed open. Since neither the probe-packet nor its potential ICMP packet are guaranteed to arrive (remember, UDP is connectionless and so is ICMP) nmap will typically send several UDP packets per UDP probed port to reduce false positives. In my experience the accuracy of nmap's UDP scanning varies among target OSes, but it's better than nothing.

RPC Scan—used in conjunction with other scan types, this feature will cause nmap to attempt to determine which of the ports identified as open are hosting RPC (remote procedure call) services, and what those services and version numbers are.

Ping Scan (Sweep)—see "Types of Scans and Their Uses" above.

Whew! Quite a list of scanning methods—and I've left out ACK scans and Window scans (see the man page on nmap(1) if you're interested). nmap has another very useful feature, OS fingerprinting. Based on characteristics of a target's responses to various arcane packets that nmap sends, nmap can make an educated guess as to which operating system each target host is running.

Getting and Installing nmap

So useful and popular is nmap that it is now included in most Linux distributions. Red Hat 7.0 and Debian 2.2, my two current flavors of choice, both come with nmap (under Applications/System and Extra/Net, respectively). Therefore, the easiest way for most Linux users to install nmap is via their system's package-manager (e.g., RPM, dselect, YAST) and preferred OS installation-medium (CD-ROM, FTP, etc.).

If, however, you want the very latest version of nmap and/or its source code, both are available from <http://www.insecure.org/> (Fyodor's web site) in RPM and TGZ formats. Should you wish to compile nmap from source, simply download and expand the tarball, then enter these commands (allowing for any difference in the expanded source code's directory-name; nmap v2.53 may be obsolete by the time you read this):

```
cd nmap-2.53
./configure
make
make install
```

Using nmap

There are two different ways to run nmap. The most powerful and flexible way is via the command prompt. There is also a GUI called nmapfe, which constructs and executes an nmap for you (see Figure 1).

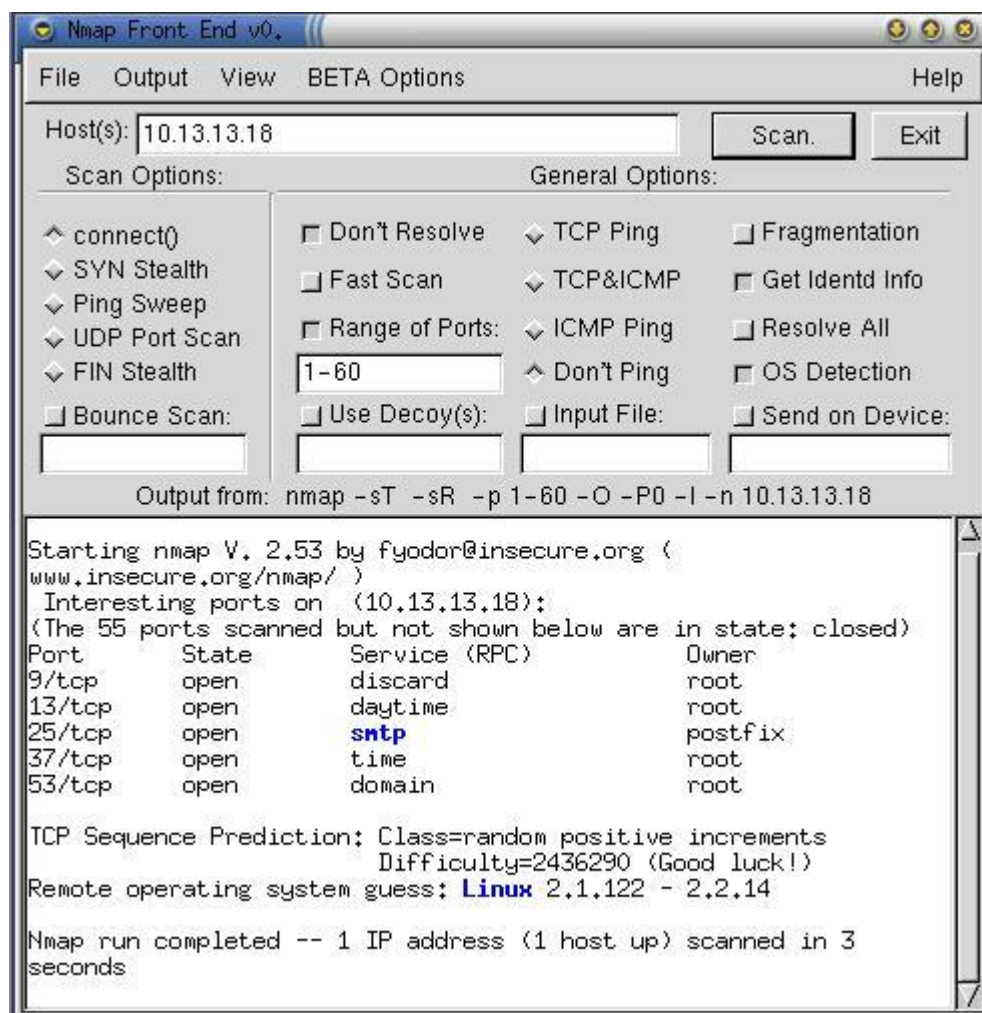


Figure 1. Sample nmapfe Session

This GUI is useful for quick-and-dirty scans or as an aid to learning nmap's command-line syntax. However, I strongly recommend learning nmap proper since the GUI presents only a subset of nmap's features. Besides, it's overkill to fire up X for little old nmap.

The nmap commands are easy to learn. The basic syntax for simple scans is: nmap -s (scan-type) -p (port-range options) (target). The -s flag is immediately followed by one of the following:

- T: TCP Connect Scan
- S: TCP SYN Scan
- F: TCP FIN Scan
- N: TCP NULL Scan
- X: TCP Xmas Tree Scan
- U: UDP Scan (can be combined with above)
- R: RPC Scan (can be combined with above)

The `-s` flag, which specifies which type or types of scan to run, can be followed by any of the TCP scan-types, U for UDP scanning, R for RPC scanning/identification or any combination of these three groups of flags. Only one TCP scan-type may be specified in a given session, however. If the `-s` flag is omitted altogether, the default scan-type is TCP Connect.

For example, `-sSUR` tells nmap to perform a SYN scan, a UDP scan and finally an RPC scan/identification on the specified target(s). `-sTSR` would fail, however, since TCP Connect and TCP SYN are both TCP scans.

If you state a port range using the `-p` flag, you can combine commas and dashes to create a very specific group of ports to be scanned. For example, typing `-p 20-23,80,53,600-1024` tells nmap to scan the ports 20 through 23, 80, 53 and 600-1024. Don't use any spaces in your port range, however.

Similarly, the "target" expression can be a hostname, a host IP address, a network IP address or a range of IP addresses. For example typing `192.168.17.*` expands to all 255 IP addresses in the network—192.168.17.0/24 (in fact, you could use `192.168.17. 0/24` instead); `10.13.[1,2,4].*` expands to 10.13.1.0/24, 10.13.2.0/24, and 10.13.4.0/24. As you can see, nmap is very flexible in the types of target expressions it can understand.

Reality Check I: Some Simple Sample Scans

Before we go any further, let's examine some basic scans that use the flags we've discussed so far. The examples in this section all use nmap version 2.53 (the most current as of this writing) running on Red Hat 7.0. The target system in these examples is running Windows98 with Sambar web server installed and active for good measure.

First, suppose we want nmap to perform an "all-default" scan. We're not required to provide any flags if we don't want to; given nothing more than a target IP or IP expression, nmap will ping each target host and then scan it using the TCP Connect method on (destination) ports 0-1024 plus all other ports listed in `/usr/share/nmap/nmap-services` (your path to this file may vary), for a total of 1,523 TCP ports. Listing 1 shows what such an all-default scan looks like when run against a Windows98 system.

Listing 1. An "All-Defaults" nmap Scan

Note that it took only two seconds to query 1,523 ports. When I said the TCP Connect method is fast, I wasn't just whistling "Dixie".

For our next example scan, suppose we want to add UDP to the mix, and while we're at it, we want to see if any of the open ports we find are running RPC

applications. Since we want to do UDP port scanning in addition to, and not instead of, TCP Connect scanning, we'll need to say so explicitly. Our command and its input will look like Listing 2.

Listing 2. Nmap Scan Using TCP Connect, UDP and RPC Modes

The -sU and -sR scans (combined above in -sTUR) go particularly well together: RPC is a UDP-intensive protocol. When nmap finds an RPC service on an open port, it appends the RPC application's name in parentheses, including the version number if nmap can make a credible guess at one.

Suppose we're looking for something a bit more specific. This could be because we have some idea of what the host is running and/or we wish to minimize the scan time. To specify which ports we want to see, we append the -p flag along with a list of ports. Commas and dashes, but not white space, may be used in this list. Listing 3 shows a scan in which we check all privileged ports (1-1024) plus a few high ports we're worried about, namely TCP 12345 and 12346 (NetBus' default ports) and UDP 31337 (BackOrifice's default):

Listing 3. Scanning for TCP and UDP on Selected Ports

Finally, because it's so easy, let's do a scan on multiple hosts. The host expression nmap accepts is even more flexible than the port expression: wild cards, square brackets (lists) and "slash/subnet-bits" notation may be used. Here's what the command would look like to perform the scan in Listing 3 on my entire test network (254 addresses, output omitted this time):

```
nmap -sTU -p 1-1024,12345,12346,31336 10.13.13.0/24
```

Intermediate nmap

nmap sports a frightening array of features designed to sneak through firewalls, avoid triggering intrusion-detection software and otherwise help the user evade detection. I feel no urge whatsoever to discuss those features here; while no doubt they have legitimate uses, I'd like to spend the remainder of this article on a few nifty features that are less obviously cracker-oriented.

Suppose you're the administrator of a large network and someone installs a server in your machine room that appears to be reachable from the Internet, in possible violation of your organization's security policy (and/or to your indignation since you weren't asked for permission). Before going on an authority-asserting rampage, you decide to first find out as much as you can about the possible risks to which your network has been exposed.

Luckily, the mystery server has an IP address scrawled on its front in purple crayon. Equally luckily, you're a righteous nmap angel of vengeance because you read "Paranoid Penguin" this month. Here are some nmap options to help you find out just what's going on.

First, what OS is this beast running? OS fingerprinting, summoned by the `-O` flag, may tell you. When you use `-O`, nmap sends packets with various TCP options set and compares the replies it receives with its OS fingerprinting database (`/usr/share/nmap/nmap-os-fingerprints` on my Red Hat 7.0 system). In my experience this feature works extremely well, except on MacOS 8 (which seems to stump it).

Next, are any of the active ports running services with root privileges? Obviously some services need this much privilege but many don't; if the web server on this box is running as root, someone is going to need talking to for sure. Use the `-I` flag to have nmap query the target's ident daemon, whose sole purpose is to tell the world which user owns each listening service.

Can we minimize the chances an overly aggressive scan will overload the target system or the network? Oh yes. The `-T` flag allows us to specify a timing mode; the options are Paranoid, Sneaky, Polite, Normal, Aggressive and Insane, in increasing degree of network-unfriendliness (based on how long nmap waits between packets and whether it sends packets serially or in batches). `-T Polite` is a good choice if you want to go easy on your target and/or network.

How do we do a fast scan that checks for likely services without scanning all privileged ports? The `-F` flag tells nmap to scan only those ports listed in `nmap-services`. In this way, we avoid ports unlikely to yield anything interesting.

Finally, is there an easy way to save our evidence of lameness as a text file? Typing `-oN filename` tells nmap to write its results to a text file. If we want nmap to use HaX0r Sp3ll1ng, we can use `-oS filename` instead ("`S`" is for "Script-Kiddie-Talk").

In Listing 4 we see the unauthorized server is accepting connections for, among other things, Secure Shell, Telnet, HTTP/SSL, LPD, X and nessus. nessus? Why, that's a security scanner. You definitely don't want internet hosts able to reach a nessus server on your network—that just happens to be the topic of next month's column.

Listing 4. Using Some Intermediate Features

As powerful as nmap is, nessus gives us a means of going a step further and probing all these listening-ports nmap has found for known weaknesses. Again,

we'll focus on using these powerful tools for good rather than evil; in the meantime, I hope you'll do the same with nmap.

[Um, What's the Port?](#)

[Resources](#)



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](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Focus on Software

David Bandel

Issue #85, May 2001

ipmenu, manedit, chemtool and more.

Training and certification. Who needs them anyway? Well, many organizations, particularly larger ones, like certification a lot. Why? Because it tells them several things: first, it tells them the candidate is dedicated enough to the profession to get certification (with or without training). Second, it tells them this candidate actually knows something about what he/she claims to have knowledge of; they don't just have to take the candidate's word. Frankly, in many larger places, the folks who do the initial screening don't have a clue about operating systems or computers, beyond pulling up a word processor or spreadsheet and using it. This should be obvious from their blind dogma insisting all correspondence be submitted in Word format, otherwise they can't figure out how to read it. Do I insist on certification? No. I actually like someone who has little or no training/experience so I can teach my way. Take it from someone who's been there, there are at least three ways to go about any task in Linux. I happen to like my way. This debate is something the folks at LPI struggle with, too—I know I volunteer many hours working with them. Ensuring that the questions are relevant, unambiguous and not biased toward any given distribution; that the correct answers are correct (grammatically and syntactically, as well as technically); and also that the wrong answers are wrong (or at least more wrong than the right ones) is a time-consuming process. So, if you have a little time and even a little knowledge, you're invited to help devise and submit questions for consideration—you don't need to be an expert. While you're at it, why not take the exams? It won't hurt and might even get you a foot in a large and otherwise closed door.

ipmenu: users.pandora.be/stes/ipmenu.html

This little utility won't make you an iptables expert, but it will help you create, view and edit iptables' rules. Based on correspondence I've had, the most difficult part seems to be the concept of separate tables for chains, depending

on where the rules in those chains work. The good part is that while it's curses-based, it's not X-based. After all, X on firewall isn't the best idea, although I recognize that under some circumstances it will happen anyway. Requires: `cursel`, `objc`, `sh`.

manedit: <http://wolfpack.twu.net/ManEdit/>

I don't personally know many folks who can write man pages. In fact, this is one area where nonprogrammers can help out. Perhaps you just want to improve grammar, spelling or add some comments of your own to existing man pages. This utility should help you do all the above and more in a nice graphical environment. Requires: `libgtk`, `libgdk`, `libgmodule`, `libglib`, `libdl`, `libXext`, `libX11`, `libm`, `glibc`.

chemtool: www.uni-ulm.de/~s_tvolk/chemtool.html

I remember in chemistry class we had to draw chemicals to visualize the bonding. Not sure I really learned anything from it, but it was required. Well, **chemtool** does all this better than I ever could. When your creation is done, you can export it to various formats, including PostScript and Xfig. Some examples and templates are included to get you started. Requires: `libgtk`, `libgdk`, `libgmodule`, `libglib`, `libdl`, `libXext`, `libX11`, `libm`, `glibc`.

Project Clock: <http://members.optushome.com.au/starters/pclock/>

This small, lightweight utility can be used to keep track of how much time you devote to various projects during the day. It can be started easily at login, then select the project to add time to as you go. Projects are simple to add, and an included report generator will show you what you need to do come billing time. Requires: `tcl/tk`, `tix`.

gfract: <http://www.iki.fi/osku/gfract/>

While not incredibly useful, this program is fun. After all, who doesn't like fractals? This program allows you to view fractals, cycle colors and other things. Requires: `libgtk`, `libgdk`, `libgmodule`, `libglib`, `libdl`, `libXext`, `libX11`, `libm`, `libpng`, `libz`, `glibc`.

MRTG Remote Data Collector: <http://pandora.sytes.net/mrdc/>

MRTG does one thing very well: graph bandwidth usage, but it doesn't track much else. To help, **mrdc** can collect and present other kinds of data for MRTG to graph. For example, **mrdc** can pass load data so you can watch a system's load over time. Or, it can graph physical memory versus virtual (swap) memory

or number of running processes to total processes. Requires: Perl, snmp on the system from which to collect data and MRTG.

Input/Output Grapher: <http://www.dynw.com/iog/>

When MRTG is overkill, or you just don't want to configure anything that simple for a bandwidth monitor, IOG might be what you need. It uses bar graphs instead of the line graphs used in MRTG and is easier to set up and run. You will need to know what your ifInOctets and ifOutOctets device numbers are, but a walk of the **snmp** tree will show that quickly enough. Requires: Perl, snmp on the system to be monitored.

OpenRealty: <http://jonroig.com/freecode/openrealty/>

If you are a realtor or know any realtors, then this software will be of interest. It claims to be simple enough for a realtor to set up, and I imagine that means technoneanderthal realtors. It requires someone to make adjustments to the index.php page, but beyond that, this is the simplest package to administer I've seen in a while. I wish realtors had something like this set up the last time I was looking for a house in the States. If you're not in the US, you might need to make some adjustments (including translations), but it would be a trivial undertaking. Requires: web server w/ MySQL and PHP4, MySQL, 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](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

Adjusting to Life in the Bazaar

Doc Searls

Issue #85, May 2001

Doc talks about the software business.

A man walks down the street
It's a street in a strange world
Maybe it's the Third World
Maybe it's his first time around
He doesn't speak the language
He holds no currency
He is a foreign man
He is surrounded by the sound
The sound
Cattle in the marketplace
Scatterlings and orphanages

—Paul Simon

Let's talk about the software business. We'll start with Microsoft. Hey, it isn't 2010 yet. If we're going to talk about the software business in 2001, we have to address the incumbent Kahuna. It isn't easy in any case, and this past February, Jim Allchin, who runs Microsoft's platform products group, made it a lot harder. In an interview with Bloomberg, he called Linux "an intellectual property destroyer" and open source a "threat" to "the American way", among other wacky, paranoid things.

This draws attention away from much slower and less newsy changes happening at Microsoft for years and away from signs that the company is noticing its markets are fully networked and, therefore, behaving progressively less like "targets" for marketing messages and disposable consumer electronics—and more like bazaars.

A bazaar is a real market: a chaotic and noisy public place where all the reciprocal abstractions of economics—supply and demand, production and consumption—are a handshake apart. In real markets, clues from customers have a lot more impact on vendors than any vendor's "marketing strategy" has on "consumers". Since marketing likes to be "strategic" and doesn't like to touch customers (that's sales' job), it becomes progressively more useless in the bazaar environment. "Consumer marketing" becomes an antique conceit, and something else takes its place; something that involves participation in markets at a much deeper, more organic level.

Something similar started to happen at Microsoft in the summer of 1996, when the company did an unusually clueful thing: they created unmoderated Usenet newsgroups for every one of their major products. Suddenly there were conversational zones where customers, employees and everybody else could say what they pleased. Some people posted suspicions that Microsoft had created a habitat for astroturfing or otherwise spinning untruths about its products (something the company has since been accused of doing in other venues). But employees jumped in and encouraged honest exchanges. They wanted to know as much as possible about how their products worked in the real world. A Microsoft executive at the time told me this was "Marketing 101" and expressed surprise that competitors weren't doing the same kind of thing.

However, Marketing 101 in those days was still mostly an academic exercise. Theodore Levitt taught that the purpose of marketing was "to satisfy the customer no matter what". For all the clues it got from Usenet and tech support calls, Microsoft continued to make crash-prone operating systems. With good reason—they sold by the zillions. Money spoke and the market approved.

Starting in February 1998 though, the market started to speak in forms other than money. This was when Salon.com published the results of a "Haiku Error Message" contest. Within minutes of publication, the winning haikus were being mailed and posted all over the world. Although it didn't win first prize, this one by Peter Rothman unleashed one of the most viral memes in the history of the Net:

Windows NT crashed. I am the Blue Screen of Death. No one hears your screams.

Thanks to the Net, everybody's screams became one huge joke. Windows became a target of mass mockery. Worse, it became a mirror in which it mocked itself. Look up "blue screen of death" in a search engine and near the top of the list you'll find a page (<http://www.daimyo.org/bsod/>) where Matt Michie (who works at Linux.com) keeps a rogues' gallery of public BSODs. He has shots of giant Las Vegas electronic billboards, ATMs, promotional

marquees, airport arrival/departure displays and casino game machines. Naturally many of the same pictures appear on lots of other sites. Memes are like that. And the bigger the bazaar, the faster word spreads through it.

At some point, the din of clues overwhelms everything else. Clearly this happened last summer when Jim Allchin went on sabbatical with a bunch of friends in a boat. Recently, he told me he learned two things on the trip that amazed him: 1) how many different kinds of things people want to do with PCs; and 2) how much trouble they have making their PCs work. He called the experience “humbling” and said he was “more determined than ever to do something about it.” Note the word ever. This means he's been trying to do something about it for a long time, which would only make sense; whatever else anybody calls him, Jim's still an old UNIX guy.

Now we see Microsoft ads promoting “five nines” (99.999%) reliability for Windows 2000 Professional. The claim may be debatable but not the fact that reliability is a concern, as we've seen Microsoft suggest previously. Why does reliability matter now?

I think it's because the bazaar is losing at least some of its appetite for PCs marketed as consumer electronics. Starting in 2000, more customers began to realize last year's PC was still fine. So were Office 98, Quicken 7, Photoshop 5.5 and lots of other software products—perfectly adequate (if not always perfectly functional).

Then, there was the repositioning of the operating system. It used to be a pretty product. Now it's just infrastructure. For more and more people, the platform that came to matter was the Net, not the PC's OS.

And then there was the undeniable success of the Net's self-ubiquitizing infrastructure, which owed far more to free and open-source software development methods (and developers) than to anything coming from software vendors whose first urge is to lock in customers. Linux is clearly part of that. There is no way Microsoft could ignore the success of Linux or the growing importance of constantly building infrastructure.

Ubiquitous infrastructural building materials are by nature commodities. You might be able to make money selling them, but the margins are elsewhere. Still, as every business comes to depend increasingly on commoditized infrastructure, including the Net and Linux, there is a strategic imperative to contribute to that infrastructure. If you're not involved in building infrastructure, your know-how falls behind, to say nothing of innovation.

Microsoft is starting to get a little bit strategic about this stuff. That's why they worked together with other companies on the SOAP and XML-RPC protocols. This is a drop in the bucket compared to what Linux community natives like Red Hat, VA Linux, Caldera and others have been doing for years. But it's the same strategy.

So what's the strategy for Linux companies, which have been in the business of building commodity infrastructure all along? We know they can sell high-margin hardware and services. But can they sell high-margin software?

Yes. They can do this by keeping the high-margin source code closed and giving away as much open-source code as they possibly can. This is what Caldera is doing with Volution. It's what Borland is doing with Kylix. It's what Turbolinux is doing with TurboCluster. There is a huge open-source side to each of those companies' product strategies—one that involves giving away a lot of stuff. But Volution, Kylix and TurboCluster are all closed-source products. None of those companies go out of their way to point out that fact, but there it is.

Traditional closed-source companies like IBM, Sun, Borland and even Microsoft are starting to adopt open-source strategies (of very different sorts, at very different speeds). Do traditional open-source companies need to move in a closed-source direction if they want to sell high-margin products to enterprise customers? So far, their answer is yes. And they all know the arguments for open source.

So, until some company risks selling open-source software to enterprises at a high margins and succeeds, selling high-margin enterprise software will remain a closed-source business, even for vendors rooted in the Open Source community.

Doc Searls is Senior Editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

The Embedded Side of LinuxWorld

Rick Lehrbaum

Issue #85, May 2001

Rick takes a customary “walk on the embedded side” of LinuxWorld Expo in New York City.

After three days of roaming some 100,000 square feet of exhibition area populated by 200 companies' exhibits and attended by perhaps 20,000 “Penguinistas”, I've certainly had an eyeful of Linux.

After letting it all sink in for a few days, I've come up with a few general observations. First of all, there were more suits this year, more glitzy exhibits and fewer of the small (10 x 10 foot) booths. Second, many more exhibitors displayed this year—the exhibit area seemed to have doubled in size since last year's New York show. Also, this year's show suggests Linux has hit adolescence. There seemed to be much more of an air of a maturing industry in comparison with last year's New York show, with companies like IBM, Intel, Compaq, Sun, SGI and HP on hand. Finally, embedded really arrived at this LWE. A common theme just about everywhere was the growing importance of Linux in “devices” and embedded systems. In fact, I'll venture a prediction that this year is shaping up to be “the year of Linux in devices”—with products like Linux-based PDAs, cell phones, webpads and set-top entertainment systems hitting the market in growing numbers.

Of the 200 exhibitors, roughly 20 identified themselves in the show guide as offering “embedded systems” products, but nearly twice that number were promoting embedded Linux products or services of one kind or another. In case you missed the show, here's a brief summary of what I found.

Accelent Systems showed their Integrated Development Platform, an embedded Linux- and StrongARM-based reference platform to help companies develop Linux-based internet appliances and other embedded devices. See <http://www.accelent.com/>.

Applied Data Systems showed four demos of their GUI-oriented embedded Linux- and StrongARM-based single-board computers (SBCs), despite having all of their booth equipment lost in transit. These included the Bitsy, a 3" x 4" SBC running embedded Linux; the Tandem, a two-headed StrongARM SBC running two independent displays off one processor under embedded Linux; the Graphics Client Plus, running Century Software's Linux desktop toolkit; and the Graphics Master running embedded Linux. See <http://www.applieddata.net/>.

Axiom showed a number of embedded and industrially oriented single-board computers in the PC/104, EBX, 3.5" and half-size slot-card form factors. Axiom's products are now supported under Linux. See <http://www.axiomtek.com/>.

BSDi had a sign at their booth that said "eBSD—the Internet Expert's Choice to Embed in Internet Applications and Appliances". There wasn't any further explanation available nor was anyone around who could tell me anything about the company's strategy with respect to the embedded (BSD) market. See <http://www.bsdi.com/>.

At Century Software's pedestal in Red Hat's pavilion, Century Software's Microwindows, ViewML and PDA software toolkit were being demonstrated on three platforms:

- The first public demonstration of full motion MPEG video under Microwindows, based on a fast framebuffer display technology known as Simple DirectMedia Layer (SDL), was running on an ADS Graphics Master SBC.
- Microwindows, the latest ScreenTop PDA suite, 802.11 wireless networking and *Doom* (the game) were all being demonstrated on a Compaq iPAQ. Century Software CEO Greg Haerr described the latest ScreenTop programs as "more user-friendly and Palm-like".
- In this third demo, an SIS motherboard was shown booting all the way to the Microwindows desktop in just seven seconds—quite an accomplishment. The secret to doing this is a combination of LinuxBIOS, the Memory Technology Driver (using a DiskOnChip Flash disk), a small footprint Linux kernel and, of course, Microwindows. See <http://embedded.centurysoftware.com/>.

Compaq was naturally showing off the exceedingly popular StrongARM-based iPAQ PDA, which these days seems to have become the preferred platform for Linux PDA software development. Just try to buy one. See <http://www.compaq.com/>.

Coventive had its LinuxWorld debut featuring a massive circular pavilion with a large number of Coventive and partner product and technology demos

distributed all around its circumference: Axis Corporation's (Japan) canD, a very small, stylish and easy-to-install Linux-based TV set-top box with a remote control, containing a customized GUI and Linux software that fits in 8MB ROM, all based on Coventive's XTinux embedded Linux operating system; a Chinese Linux PDA manufactured by Legend, based on an Intel StrongARM processor and Coventive's XTinux, including a full-featured browser with PIM/SYNC functions as well as handwriting recognition for simplified and traditional Chinese; a Linux-based, on-line stock trading software application demonstrated on Coventive's set-top box reference design; one of the first Linux-based smart card readers from Disonic (Taiwan) and a "Gigabyte Server Appliance" with easy-to-use GUI and containing Coventive's Linux OS embedded in ROM, "allowing non-techies to set up and monitor the network environment". See <http://www.coventive.com/>.

DevelopOnline showcased both its on-line capabilities and its partners in five demos: the STMicroelectronics STPC system-on-chip Internet Appliance reference design, a demo of the DevelopOnline integrated development environment (IDE), shows precisely how developers can use the service to upload and test software on selected reference platforms using one of several on-line embedded Linux software development kits (SDKs); RidgeRun, a DevelopOnline partner, explained and demonstrated DSPLinux, an embedded Linux OS specifically tuned to the requirements of TI dual-core processors that combine ARM and DSP processors on a single chip; Insignia Solutions' Jeode Embedded Virtual Machine (EVM), a Java-compatible embedded software environment demonstrated on a ThinkPad running Red Hat Linux; and a remote demo, accessed via DevelopOnline's web technology, shown running the Jeode EVM and the PointBase embedded database running on an iPAQ under Hard Hat Linux. See <http://www.developonline.com/>.

Hewlett-Packard demonstrated the new Chai Appliance Platform for Linux, a suite of integrated software components for creating internet-enabled information appliances. See <http://www.embedded.hp.com/>.

IBM showed their ever-popular Linux wristwatch, a complete Linux system in the body of a tiny (56mm x 48mm), geeky looking watch. IBM was also demonstrating DB2 Everywhere, a database for embedded systems that fits in less than 150KB memory, which was announced at the show. In another part of their very expansive booth, IBM showed a "technology demonstration" of a Linux-based retail point-of-sale (POS) system. See <http://www.ibm.com/linux/>.

Intel showed several demonstrations of embedded Linux, including the Assabet reference design for the StrongARM SA-1110 (which forms the basis of Compaq's iPAQ design). The Assabet was running from a battery pack and, according to the Intel spokesperson, achieves 10-12 hours of operation with the

LCD backlight turned off, or roughly six hours with the backlight on. Intel's new XScale microarchitecture, the successor to StrongARM, was also being demonstrated running embedded Linux at the MontaVista and LynuxWorks booths. See <http://developer.intel.com/>.

Lightning Instrumentation showed a miniature embedded Linux-based router device, the MultiCom Ethernet II. Lightning-Linux, the embedded Linux OS used within the MultiCom router device, is also being made available independent of the hardware. Much of Lightning-Linux is apparently available under the GPL. See <http://www.lightning.ch/>.

linAXE Project had a tiny booth displaying something brand new. linAXE started out as an effort to develop a Linux-based RTOS to control the popular "BattleBots" fighting robots. Since then, the linAXE Project has broadened its scope. See <http://linaxe.sourceforge.net/>.

Lineo had, as usual, a pavilion bustling with activity and providing many interesting demonstrations:

- The successor to Lineo's uCsim is the uCdim. The new soDIMM form factor (1.7" x 2.7") SBC is based on a DragonBallVZ microcontroller and, not surprisingly, runs uClinux.
- Trolltech (a Lineo partner) was present, demonstrating Qt/Embedded and the Qt Palmtop Environment (QPE).
- Rappore (a Lineo partner) was present, demonstrating an embedded Linux-based Bluetooth stack in a demo that consisted of a dollhouse with lighting, garage door, etc., controlled by a notebook computer with everything interconnected via Bluetooth wireless communications.
- The Lineo SecureEdge VPN router platform was demonstrated.
- A small PowerPC-based PC/104 form factor single-board computer from Embedded Planet was used to demonstrate M-Systems' latest DiskOnChip embedded Linux driver.
- The new Lineo Academic Student Kit was on display. This kit, which contains a uCsim, the uClinux OS, a small experimenter board and a detailed instruction manual, will be sold to students for \$250. It includes courseware and software CDs. Thirteen universities are already using the kit as part of their computer science classes. See <http://www.lineo.com/>.

LynuxWorks demonstrated the open-source BlueCat Linux and the proprietary LynxOS real-time OS (RTOS) in their pavilion, which now represents the combined strengths of LynuxWorks and ISDCorp, a company acquired by LynuxWorks last summer. Demos included an interesting display of the Linux application binary interface (ABI) compatibility of a preview version of the next release of the proprietary LynxOS real-time OS (RTOS). Similar Linux- and

LynxOS-based systems (PCs) ran identical binary images of *Quake*, along with identical images of another resource-hogging program. Real-time priority was then adjusted on the LynxOS RTOS system to show off the capabilities of “a true RTOS”. The new SpyKer real-time event profiler and trace tool was demonstrated, as was PhatNoise's Phatbox, an automobile MP3 player-based on embedded BlueCat Linux. A high-availability chess game, based on BlueCat Linux with host failover, was demonstrated at Intel's booth, and M-Systems displayed DiskOnChip support for BlueCat Linux. See <http://www.linuxworks.com/>.

Metro Link demonstrated Micro-X, a windowing solution for embedded Linux. “Micro-X is based on the X Window System protocol, so you won't have to learn a new interface to develop for embedded systems”, they say. Micro-X was shown running on an Intel StrongARM SA-1110 platform. Micro-X fits in as little as 575K of memory and supports x86, PowerPC and ARM/StrongARM, and will soon support MIPS. See <http://www.metrolink.com/>.

Metroworks showed their recently enhanced CodeWarrior IDE that provides embedded debugger and remote debugging capabilities on a Motorola PowerPC 8260 reference board and an Embedded Planet PC/104 form-factor PowerPC 823 SBC. The booth's theme was “Embedded Linux—Increasing the IQ of Smart Devices”. See <http://www.metroworks.com/>.

MontaVista had lots to see, with eight product demos and five partner demos. Some highlights follow:

- A high-availability CompactPCI system demo-based on Hard Hat Linux, Ziatech and Motorola CompactPCI processors, with MontaVista's High Availability software, backplane networking software and hot-swap drivers.
- Various cross-development tools, graphic embedded kernel configuration tools, graphical IDEs, remote debuggers and performance analysis tools.
- **@win**, a small footprint (200KB) GUI/windowing system from MontaVista partner Adelinix (Korea), running on a Compaq iPAQ (compatible with GTK/GTK+ and QT toolkits).
- A Compaq iPAQ PDA running Hard Hat Linux and the X Window System with internet browsing via Netscape 6 and live video display within an X window.
- MontaVista's approach to a hard real-time Linux kernel using two identical systems running sound applications. One had a “vanilla” Linux kernel, while the other contained MontaVista's real-time enhancements. The CPU and scheduling latency for each was shown on a continuously refreshed chart.

- Motorola's 74xx AltiVec processor running Hard Hat Linux on a Motorola Sandpoint embedded reference design, support for embedded Java applications under Hard Hat Linux based on IBM's VisualAge Micro Edition (VAME) and IBM's 750CX/CXe PowerPC running Hard Hat Linux.
- A StrongARM-based customer retail point-of-sale subsystem (designed by RadiSYS for USA Technologies) called ePort was shown as an exposed electro-mechanical subsystem and mounted inside a soft drink machine, so now you can browse the Web while you buy your Coke.
- A customer's 1U "industrial strength" rackmount gateway appliance (made by Diversified Technologies) with Hard Hat inside.
- One of the first working demonstrations of embedded Linux (Hard Hat Linux, of course) running on Intel's new XScale microarchitecture (the next generation of the StrongARM processor) running on Intel's XScale evaluation platform. See <http://www.mvista.com/>.

Neoware's award-winning Eon computing appliance (the Anything Box), running the Neolinux 2.0 embedded Linux operating system, was demonstrated. Unique internet appliance-oriented features such as ezConnect (a simple user interface) and ezSnap (a software distribution capability) were demonstrated. See <http://www.neoware.com/>.

NexCom showed a number of embedded and industrially oriented single-board computers in a variety of popular form factors, large and small. NexCom's products are now supported under Linux. See <http://www.nexcom.com/>.

OnCore Systems demonstrated OnCore Linux for Real-Time, a real-time platform that can host multiple copies of Linux in a hard real-time system environment. See <http://www.oncoresystems.com/> for details.

PEP Modular Computers showed their 3U and 6U CompactPCI boards and systems now supported under Linux. These kinds of products are used in high-reliability systems for telecom and internet infrastructure, industrial control, military and medical applications. See <http://www.pep.com/>.

Portwell showed a number of embedded and industrially oriented single-board computers in EBX and other embedded and industrial form factors. Portwell's products are now supported under Linux. See <http://www.portwell.com/>.

Red Hat showcased three examples of embedded technologies: the Century Software discussed previously; the Red Hat Embedded Linux Development Kit running on a Motorola MBX860 EBX form-factor single-board computer; and uClinux and eCos running on a NetSilicon NET+Lx reference design system. See <http://www.redhat.com/embedded/>.

RedSonic showed the RedIce-Linux real-time Linux operating system and associated development and debugging tools. Additionally, RedSonic showed off its hardware/software set-top box real-time and embedded-system reference design. See <http://www.redsonic.com/>.

Sun Microsystems showed several demonstrations of the J2ME Java for embedded devices that supports embedded Linux-based systems. Hardware platforms included the Compaq iPAQ PDA and an Arcom embedded Linux reference platform. The J2ME occupies between 2 and 4MB of memory space. Wireless 802.11 technology was also demonstrated on the iPAQ embedded Linux system. See <http://www.sun.com/software/>.

TimeSys showed five demos of real-time Linux that demonstrate the quality-of-service (QoS) and real-time capabilities of TimeSys' Linux/RT operating system:

- An Embedded Planet PowerPC 860 board running TimeSys Linux/RT, with TimeTrace used for dynamic visualization of system events and context switches.
- JTime, the TimeSys real-time Java Virtual Machine, running on top of TimeSys Linux/RT, controlled a small robot via a wireless serial connection.
- A StrongARM-based Corel NetWinder system demonstrated Linux/RT's QoS capabilities through the reservation of computation time on behalf of two chess programs.
- The QoS capabilities of Linux/RT on a desktop system, by assigning network and CPU reservations for video and audio conferencing software, demonstrated its capability to maintain acceptable live audio/video performance despite heavy system loading.
- *Quake III* was shown running on a Linux desktop with CPU reservation controlled by a unique "QoS knob" that altered the QoS setting—you could turn the knob clockwise/counterclockwise and watch the game's performance get better/worse right before your eyes. See <http://www.timesys.com/>.

Transvirtual Technologies demonstrated the latest version of PocketLinux, a Linux/Java implementation for PDAs, handhelds and embedded computers. Interesting new features shown included 802.11 wireless communications, plus MPE and MPEG media players. Many PIM applications are now available for PocketLinux, including both utilitarian programs (calendar, address book, memo pad, e-mail, Flash player, instant messaging, XML/XHTML browser, etc.) and tons of games. See <http://www.transvirtual.com/>.

Trolltech demonstrated Qt/Embedded and the Qt Palmtop Environment (QPE) running on a Compaq iPAQ PDA. They were also giving away free floppies

containing a PC-bootable embedded Linux system consisting of Qt/Embedded, PIM applications (date book, address book, text editor, file browser, etc.) and games (Mine Sweep, Tetrix and Solitaire). The floppy image is downloadable from Trolltech's web site. See <http://www.trolltech.com/>.

Viosoft showed their embedded Linux cross-platform SDK for RISC platforms, including MIPS, ARM/StrongARM and PowerPC. They also demonstrated Arriba!, a Java-based IDE with a Windows-style graphical debugger. See <http://www.viosoft.com/>.

ZF Linux Devices showcased support for their MachZ system-on-a-chip, including the tiny MachZ demo board, Integrated Development System (a MachZ-based internet appliance reference design) and Tri-M's MachZ-based PC/104 form-factor SBC, featured in *Embedded Linux Journal's* design contest [see "Hack Embedded Linux for Fun and Prizes" in the November 2000 *Embedded Linux Journal* supplement]. See <http://www.zflinux.com/>.

As you can see, there were lots of embedded products and demonstrations at LinuxWorld—many more than at the previous shows. At the rate Linux is penetrating the embedded market, we can expect to see an even greater amount of embedded Linux activity at the next US LinuxWorld Expo and Conference, to be held August 27-30 in San Francisco, California.



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.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

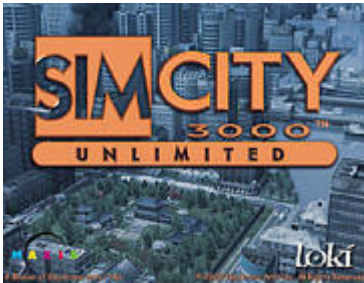
[Advanced search](#)

SimCity 3000 Unlimited for Linux

Neil Doane

Issue #85, May 2001

These top-down, huge-scale simulated world games could be laced with narcotics and not have any more appeal, and *SimCity 3000 Unlimited* is the best of the best in a game genre it helped to create.



- Manufacturer: Loki Entertainment Software
- E-mail: sales@lokigames.com
- URL: <http://www.lokigames.com/>
- Price: \$49.95 US
- Reviewer: Neil Doane

It seems that almost every person who enjoys computer-based games has, at some point, sat down with a copy of some version of *SimCity* and tried it out. For some, the lure of godlike power over an entire city paled in comparison to the blood and gore of their favorite first-person shooters. (It does take a certain someone to try to organize the myriad workings and economics of an entire city when they could just as easily run around blasting bad guys with plasma rifles in a space arena somewhere.) Some of us, however, have sat down with a *SimCity* variant only to snap out of our game-induced stupor half a day later, finding ourselves the mayor of a city with a quarter-million happy citizens. For us, these top-down, huge-scale simulated world games could be laced with narcotics and not have any more appeal, and *SimCity 3000 Unlimited* is the best of the best in a game genre it helped to create.

The Story

SimCity 3000 Unlimited is the latest installment in what has become Maxis' flagship series and is the first to make it to Linux. There is no real story line, per se, though it is playable in short bursts in games with specific goals. As we'll talk about in a minute, the most powerful *SimCity 3000* addictions come from its prowess in open-ended games that allow players to build an entire metropolis completely from scratch. You start with a little money and a plot of land that you get to modify to your own personal specifications. Then you start creating your city, zoning areas for the desired type of development in the area, implementing an array of complex civil engineering feats and performing various city management duties, like balancing a budget, implementing taxation on your citizens, passing ordinances, etc. As you get more money from taxation, your city grows organically (you get more money, you expand your city in a way that attracts people, more people move in, you get more money), and through dealings with other nearby cities, you can create relationships to facilitate your city's growth. In this mode of play, the game is basically done when you think you've accomplished all you can with the design you've chosen. Then you go back and start again, using lessons you've learned from previous cities to make your new city even better.

New for *SimCity 3000 Unlimited* is the ability to again play "scenarios", a feature found on *SimCity 2000* but notably absent in the first release of *SimCity 3000* (the non-Unlimited version). These thirteen new scenarios basically put you in a specific situation in a prebuilt city and challenge you to use your expert leadership skills to save the day. For example, in the "Criminalville" scenario, mobs haven taken over Moscow, and it's your job to sort out the mess and give them what they want. In "Fall of the Wall", you must connect East and West Berlin and "bring balance to the city". Some of these scenarios are quite easy and can be played in 45 minutes to an hour, while others will have you biting your lip in frustration at 4 **A.M.** on a Tuesday.

Game Play

As one might expect from a game whose focus is on minute fiscal and physical management of a huge metropolis, *SimCity 3000 Unlimited's* game play may seem complex at first. If you've played a *SimCity* version before, the overall feel will be quite close to that, though there are some evolutionary interface enhancements that will seem natural to a veteran. If you've never played *SimCity* before (what planet are you from again?), the controls may seem a bit much at first. However, they will become as natural as breathing with minimal effort on your part. Maxis has gone to great pains to make *SimCity's* interface as user-friendly and intuitive as possible.



Figure 1. Maxis has gone to great pains to make the interface as user-friendly as possible.

At the start of the game, you're able to model the approximately 100 square miles of landscape to your own personal specifications and to a high degree of detail. No longer are you limited to simply settling for a randomly generated map; with *SC3U*, you can create a random map or edit every feature (from mountains, to lakes, to trees or anything else) of a map of your choosing, or you can opt to use a map based on the topographies of several famous cities, such as Bonn, London and Boston. You can also opt for the types of terrain you want your selected landscape to be placed on (from desert to snow-covered), what style of buildings will be built and even which of the five available varieties of plant life will grow in your world. The grid's appearance is *highly* customizable; not only can the basic types of buildings be chosen, but *SC3U*'s Building Replacement Manager allows you to swap out old-style buildings you don't like for new ones of your choosing. Don't like any of the seemingly endless array of prefabricated buildings available to you? Use *SimCity 3000*'s Building Architect Plus to modify existing buildings or create your own complex structures from scratch using BAP's 3-D building creation environment. Remember now, this isn't the game, just the options available to *play* the game. While this level of complexity may not appeal to everyone, it's nice to know it's an option if you want it.



Figure 2. Total Control: SC3U's Terrain Editor

The game itself is played continuously (though it can be paused) as your city grows. You create and control everything from transportation infrastructure (subways, trains, roads, airports, seaports, etc.) to the placement of landmarks, from outfitting and maintaining the police department to managing your city's water supply and running appropriate plumbing to your neighborhoods. You're in charge of controlling pollution and dealing with the city's garbage problems. You must zone your city appropriately to attract a balance of residents with industry for them to work in and commercial businesses for them to spend their hard-earned cash in. (There are even three types of each of these zones—low, medium and dense—corresponding to the types of businesses/residences you're looking to build in a given area.) You still control taxation, but this is no longer your only source of income; you can pass ordinances that will make your city money (allowing gambling and casinos, for example) and/or generally make your city a better place. You can also make special trade agreements with nearby cities to put money in your pocket. For instance, is your water system more than adequate for your city's current needs? Why not sell some of that water to a neighbor for a while? To keep the mayor apprised with the latest information as the game progresses, you're kept up to date by a scrolling ticker along the bottom of your screen, which is actually a refreshing change from the somewhat irritating newspaper-style information windows that would occasionally pop up in earlier versions. You also have a staff of advisors who freely offer their opinions to you from time to time and who are available to consult with when you are uncertain about how to proceed.



Figure 3. Consulting with Advisors and Balancing a Budget in Tokyo

The *SC3U* graphics are superb. Each building is actually created in a 3-D environment with painstaking detail. You'll find more than 400 different varieties of buildings (hundreds more than previous *SimCity* versions). What's more, if you decide halfway through the game that you want to switch on the fly to a completely different style of architecture (say, from European to Asian), you can do it with the press of a mouse button. You can even build any of the dozens of included real-life landmarks like the World Trade Center, United Nations building or the Eiffel Tower to spruce up your city's skyline. Also, the in-game animations are just awesome. The Thanksgiving parades are optional, but when you choose to have them, zoom in for a close view and what you'll see is nothing short of amazing; full screens of people, marching bands, shriners in little cars, floats and even those huge inflated parade balloons; all taking their time to meander down your streets from one end of town to the other. This level of attention to detail is really one of the main reasons *SC3U* looks so good.

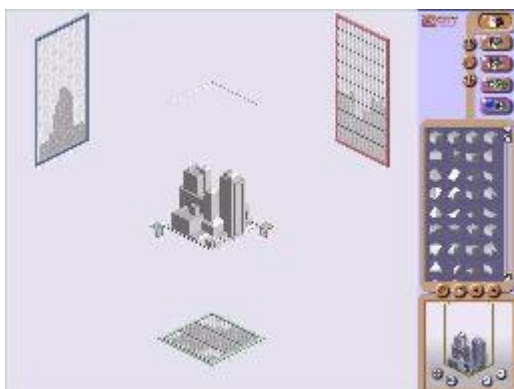


Figure 4. Create Your Own Structures with the Building Architect Plus

Find that your city is looking just *too* good and you want another challenge? Why not unleash one of the many canned disasters on your city and then try to clean up the mess? There are four new disasters in *SC3U* not found in earlier versions, as well as a great collection of old favorites, from riots to plagues of locusts to Armageddon-like space debris. If you want to make it interesting (or are just feeling destructive), the fine folks at Maxis have given you the tools to do it.

The music is, well, it's Maxis—you sort of expect the music to be a bit weird. It's soft and soothing and prone to put you in a trancelike state of mayoral bliss for extended periods of time, but it *can* get old after a while. I found myself switching off the computer speakers to help me concentrate from time to time. Again, this isn't a cut on the music per se, it is well-done music, but *any* music would probably start to get old after the half-days of gaming that *SC3U* can induce.

Linux Notes

I tested this on a G400-based VA workstation with 128MB RAM and a 500MHz PIII as well as a GeForce-based system with 256MB RAM and an 550MHz PIII, and both systems handled the game with little difficulty. Though the game doesn't require any 3-D support, the calculations required for simulating a 100-square-mile city can be rather taxing, and once you get to higher levels of development, the slowdown in movement around the screen and zoom redraws start to become more and more noticeable. Loki lists the minimum hardware requirements as a Pentium 233 and 32MB of RAM; while these numbers may make sense for many small-scale cities, I think more advanced users (or users of some of the truly monstrous scenario cities) might find it nearly impossible to construct massive cityscapes with those computational limitations. Loki recommends that you run this game in 16bpp color mode to eliminate the on-the-fly conversions that must occur if you use 24 or 32bpp color modes. If you want to use the Building Architect Plus tool, you'll need to use an OpenGL-compliant 3-D driver of some sort (both my NVidia OpenGL drivers and Mesa had no problem at all with the BAP tool). You'll want to have an OSS-compatible sound card and a minimum of 450MB of disk space for the install (a full install will run up around 650MB or so). In general, the requirements of this game really are pretty low compared to the return you get in game play. I've run this gem on my little 500MHz laptop often and find it is just as playable there as on any workstation.

Summary

It's the best version of *SimCity* yet, with more levels of customization, more add-ons, more options, better graphics and better game play than ever before. The game is, perhaps, somewhat complex by its very nature, and if you're looking for a simple game, this probably isn't your bag. However, if you enjoy the idea of some planning and strategy, and if you enjoy a fun mental challenge that is hugely rewarding, this is your game. In my opinion, this is probably a must-have for most Linux gamers.

[The Good/The Bad](#)

[Just the FAQs Please](#)



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

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

A Call to Action

Leslie Proctor

Issue #85, May 2001

On Linux International, SIIA, the Gnome Foundation and the Linux Professional Institute.

I live in Los Angeles, where the memory of the Hollywood 10 and the Blacklist lives on. I knew many of the blacklisted writers from my Sundance days and the experiences they had with HUAC left them bitter and sometimes broken. These were incredibly talented people whose lives were ruined by “The Big Lie”.

While it's easy to poke fun at remarks from Microsoft's Jim Allchin—and yes, there was a lot to laugh at—it also raises a flag. It's easy to dismiss his seemingly ridiculous charge that free software is un-American, but I'm not sure we should just laugh it off. There's always a chance that some uneducated loose cannon will take Allchin's remarks at face value, especially if the loose cannon happens to hold a public office somewhere.

GNU/Linux companies and people who are passionate about Linux need to make their voices heard on Capitol Hill and in local and state legislatures. Linux International (<http://www.li.org/>) has changed its bylaws recently to include the ability to form special interest groups and lobby. SIIA (<http://www.sii.net/>), the oldest and largest trade association of the software industry, hosted an Open Source Division meeting at their annual meeting (see full item below) with the idea of forming an agenda to address with legislators.

While flaming easy targets like Allchin is personally gratifying, we also have to be very vigilant—utilizing the existing frameworks of .orgs like Linux International and SIIA warrants a closer look.

Profile—The Linux Professional Institute

The Linux Professional Institute (www.lpi.org) designs and delivers a standardized, multinational program to certify levels of individual expertise in

Linux. The programs are put through a rigorous benchmarking process so that they will satisfy the requirements of Linux professionals and the organizations that would employ or contract them. They currently have Level 1 certification available for basic level system administrators and are working on Levels 2 and 3. LPI maintains vendor neutrality, and the testing is deployed globally, in English, through Virtual University Enterprise (<http://www.vue.com/>).

“The Linux Professional Institute preserves freedom of choice in how people prepare for their certification exams,” Dan York, president of LPI said. On their web site they offer a list of books, training centers and courses that can help someone prepare. LPI developed the exams using peer industry standard practices and psychometrics. In other words, they polled more than 1,400 people for the Level 1 test, asking them to complete a job-task analysis survey. The survey assesses tasks and knowledge sets that a person at each level of proficiency should know how to do. The goal, according to York, is to develop objectives that can be validated on a real-world basis. Legal defensibility of the tests is also an important component.

LPI is currently developing the Level 2 test and has polled more than 200 professionals thus far. The task is time consuming and lengthy, but creating a solid testing metric is worth the time and effort, according to York.

LPI could use the help of Linux professionals in the exam process. They need people who can review, write and translate Level 2 questions, as well as do ongoing maintenance and review of the Level 1 test to help keep it current. They could also use a hand in fund-raising and marketing. Because they have no budget for marketing, participants in the certification process learn about LPI through word-of-mouth. Companies could consider giving employees a few hours a week to work on LPI initiatives. For further information, log on to <http://www.lpi.org/> and click on the Getting Involved button.

SIIA Holds Open Source Division Meeting in San Diego

The Software Information Industry Association (SIIA) held the inaugural meeting of the Open Source Division, a new special interest group at their annual conference in San Diego. The agenda included discussions on the future of the open-source computing model and strategies to increase market acceptance of free software. They also discussed public policy developments affecting open-source companies and created a strategy for future action. Red Hat was the first free software company to join SIIA.

The SIIA is an international trade organization for software companies. During the antitrust case of recent history, SIIA filed a brief in the case siding with the government. This was done despite a Microsoft executive sitting on SIIA's board of directors. Within days of the filing, Microsoft resigned its membership.

According to Ken Walsh, president of SIIA, it was more important to do the right thing and go with the majority of the board's opinion, than to bow to one member.

GNOME Foundation

The GNOME Foundation (<http://www.gnome.org/>) released GNOME version 1.4, a mature, stable release of the desktop environment. The release includes a host of cool bells and whistles, as well as some additional packages, including Nautilus, Eazel's file management system. The release represents a big step forward for GNOME, moving more solidly in the usability and power category.

GNOME 1.4 has extensive documentation available through the Help tab. It is available for download on a number of mirror sites that can be reached through <http://www.gnome.org/> and will be available in many distributions. Check the web site to get information on the most current ways to get GNOME 1.4.



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.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

The PC Weasel 2000

Jon Valesh

Issue #85, May 2001

The PC Weasel is an ISA card that replaces the system video card in servers intended for headless (without monitor and keyboard) operation.



- Manufacturer: Real Weasel
- E-mail: info@realweasel.com
- URL: <http://www.realweasel.com/>
- Price: \$250 US + shipping
- Reviewer: Jon Valesh

If you ask system administrators the benefits of Linux as a server, one ability that always seems to come up is remote access, or the ability to get into the server to do anything that needs doing, without needing to be where the machine is. Linux—UNIX in general—handles that wonderfully. Unfortunately, PCs don't. To access CMOS settings or watch a system boot, you need to have a display attached, and you need to be there watching. There is nothing Linux, or any operating system, can do about it.

But the PC Weasel can help. The PC Weasel is an ISA card that replaces the system video card in servers intended for headless (without monitor and keyboard) operation. It acts as a video card and, with the connection of a short

cable between the card and the system's keyboard port, as a keyboard too. With the PC Weasel, the monitor and keyboard are replaced by an RS-232 connection and any standard terminal emulation (VT100 or ANSI) running on anything from a dumb terminal to a handheld computer to another server.

That isn't all the Weasel can do. On motherboards with reset connectors, the PC Weasel can provide remote reset control that allows you to reboot locked servers or disable computers that may be causing problems on your network. There is also a watchdog timer mode, allowing the PC Weasel to reset the system automatically if the OS locks up, or it at least stops telling the card that everything is okay. Watchdog support requires kernels with PC-Weasel watchdog support, and you may need to patch your kernel to get this.

Having a serial console on a PC provides many advantages, not the least of which is that you can control a lot of computers from one workstation by using multiport serial cards or inexpensive RS-232 switch boxes to connect all of the headless servers to the workstation. This can save effort and time if you have a lot of rackmounted PCs. But, far more valuable, the PC Weasel allows you to place servers in remote locations and provide true remote administration. You can connect a standard external modem to the PC Weasel and have dial-in access to the computer—not just the running operating system, but the whole machine. You can watch the system self-test, change BIOS settings, view POST codes, diagnose failed hardware and do a few other jobs that usually mean having someone present with the computer.

And the PC Weasel isn't limited to Linux. It will work with any OS that supports text-mode video, including DOS and most UNIX variants.

Documentation

The PC Weasel manual is a spiral bound mini-tome of engineering wit and detail—a lot of detail, including everything from how to set the board jumpers to compiling custom code to run on the PC Weasel's on-board microprocessors. Though it tends to jump into low-level details with both feet, there is enough general information that anyone with a fair tolerance for jargon can get what they need. Given that the PC Weasel is aimed at a traditionally tech-savvy user base (system administrators, technical support personnel, embedded systems developers, etc.), the documentation is probably exactly right: human enough to be fun to read, but technical enough to be precise.

These days, when it seems like everyone has ADD, a lot of people (me, for example) would benefit from a single-page reference guide with nothing but the few facts most people would need who use the PC Weasel card. The card is fairly self-explanatory, but such a guide would be nice.

Installation

First the limitations: you need an available ISA slot; the PC Weasel doesn't come in a PCI version. Second, you can't have another video card installed in the computer alongside the PC Weasel. Given that, installing the PC Weasel is fairly simple, provided your computer's motherboard is willing. An unfortunate number of modern motherboards have video controllers built in, and unless the on-board video disables itself properly when the PC-Weasel card is installed, you may have problems. The PC Weasel's keyboard connection is made by way of an external jumper cable that is just a little bit messy and prone to being yanked out by careless people working around the computer. The reset feature requires that your motherboard have a reset jumper, a feature that a few new motherboards seem to be skipping.

The most painful part of installing a PC Weasel may very well be finding an ISA slot and the reset jumper on your motherboard. After removing the old video card, inserting the PC Weasel and hooking up the various cables and connecting your RS-232 cable to a PC or terminal, everything pretty much works.

The PC Weasel defaults to a safe 9600 baud communications rate, adjustable once a terminal is connected. The configuration is stored in nonvolatile EEPROM, so you don't have to worry much about forgetting your settings.

How Well Does It Work?

Before I get into how well it works, here's a few words about how it works. It is important to note that in many ways the PC Weasel is an independent computer that talks to the system through some fairly narrow channels, and the narrowness of the connection provides for rather narrow areas of dependence. The first and most basic of the dependencies is power: the PC Weasel draws power from the system motherboard, and as long as the system motherboard has power, the PC Weasel runs. When the system is powered on, the PC Weasel boots separately, and runs separately, from the system CPU. This has some very positive implications for remote administration because it means that you can connect to a system even if the motherboard has completely locked up, as long as there is power. You'll only see the last screen the system displayed before it crashed, but at least you'll see something, and because the PC Weasel has abilities that go beyond a display and keyboard, you can perform a hard reset of the locked-up computer and then watch it boot without disconnecting and having to wait, blind and hoping, for the system to start.

Anyone who has ever had to call a remote office, data center or business to ask them to reboot a server, and has had to wheedle and beg to get someone to

spend the five minutes it takes to find the (hopefully) right server and press the reset button, will appreciate being able to perform their own rebooting and watch it happen.

That said, there are some limitations: you get nothing but text. Forget about boot graphics or fancy boot loaders; absolutely forget about X or using any operating system that has forgotten how to think in ASCII; text is what you'll get, so save the graphics for after the OS has booted, when you can use OS-level remote access.

And the necessary kludges: a terminal doesn't have all the keys a 104-key keyboard has. And even if it did, it probably wouldn't have a Remote Reset Button key, or a Configure the PC Weasel Card key. The solution is to use an "attention" keystroke to bring up PC Weasel-specific menus, similar to many Telnet applications. The default attention keystroke is Ctrl-^ or Ctrl-~ (depending on your terminal), though that can be reconfigured. When you send the attention keystroke, a command prompt appears at the top of the screen. From there, you can type a single-character command, or a multicharacter mnemonic that represents one of the keys not available on most terminals. For example, when you are in the attention mode, typing **xlshift** is the same as pressing (and releasing) the left Shift key. The command **xscrlk** is like pressing the Scroll Lock key. This is somewhat typing intensive but allows you to access all of the keyboard functions. It is also fairly unimportant when dealing with operating systems like Linux; as it's designed to be terminal-friendly, you rarely need to go beyond the keys a terminal would have anyway. There are also special sequences for using the Ctrl, Alt and Shift keys as modifiers, so you can, for example, send a three fingered salute by typing a command like **xc-a-del**.

The most valuable application of the PC Weasel is in remote servers. Whether you have a web server colocated in an ISP or a data logger on the side of a mountain, having to drive hours—or worse, wake someone up so they can unlock the site to let you in—to get to your server can be a killer. Having another way into your equipment can be a lifesaver. Most of us count on SSH or Telnet to get us in, but when the normal communication paths go away, being able to make contact could save you the price of the PC Weasel the first time something went wrong.

To make the PC Weasel even more useful, it has another feature for those who need to move data as well as control the machine: the card can be used as a standard serial port by the PC in a passthrough mode where data coming from the serial port is passed over the line that would normally provide screen and keyboard information. How is this useful? If you are setting up a remote data logger or need to be able to move information, not screens, you can have system software use the PC Weasel's serial port and carry out all

communications through a single modem or serial cable. I didn't try this, but it should also be possible to connect to a computer using the PC Weasel's serial console and start a PPP connection over that serial port, allowing both core system access when something has gone wrong and TCP/IP for the more flexible access it provides.

Within the limitations of its hardware and the emulations it connects to, the PC Weasel works very well.

For Remote Embedded Systems People

Okay, you aren't a system administrator. You don't care about web servers. You do care about real-world solutions, and those solutions involve putting a PC somewhere in the real-world. And, in the real world, that means that every once in a while one of the people you sold your real-world solution to calls and asks, quite reasonably, why it isn't doing anything. You have a few weapons in your arsenal to answer those questions. Maybe you have **pppd** set up waiting for an incoming call on a modem line, maybe **getty** is good enough, maybe you've been selling these systems so long that you've got DOS and a five-year-old copy of Closeup. Whatever you have, it works, but all too often you've got to call your customer and have them go over and do something to the system you sold them. The PC Weasel, out of the box, can do a lot toward making those calls disappear: the watchdog timer will hide persistent problems, and the reliability of not counting on remote-access software loaded after the OS will keep you from needing to call and ask someone to kick the hardware (or worse, sending someone driving/flying to do the same job), but the PC Weasel can be made to do more. By writing your own software to run *on* the PC Weasel, you can make the Weasel smart about the problems you face. For example, you could change the watchdog timer to automatically call your pager to let you know something went wrong rather than simply resetting the computer. Or if you want to get fancy, you could write software to watch the screen automatically for specific messages and type preconfigured commands when they are seen, perhaps restarting your crashing software or performing a soft reboot rather than an imperious hard reset. The PC Weasel includes a software API and information (including sample code snippets) on writing custom software to run on the card, independent of the system CPU.

Summary

The PC Weasel adds to the capabilities of a standard PC running any text-mode operating system, providing a watchdog timer, remote reboot capability and a serial console to make bringing large numbers of server consoles to one workstation or remotely accessing those same servers in a cost-effective manner. However, it may not be the best choice for everyone. At \$250 each, you need to take a hard look at how much the card will save you. How often do

your servers crash? How much effort and time is involved in hooking a keyboard and CRT up so you can watch them reboot? If the server is at the other end of an airplane flight, the PC Weasel is a no-brainer, but if the server is at the other end of the room, or if pressing the reset button is a telephone call away, it may not be the right choice.

Born at the beginning of the microcomputer age, **Jon Valesh** (jon@valesh.com) has pushed and been pushed by computers his entire life. Having run the gamut from games programmer to ISP system/network administrator, he now occupies himself by providing technical assistance to ISPs and small businesses whenever his day job doesn't get in the way.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

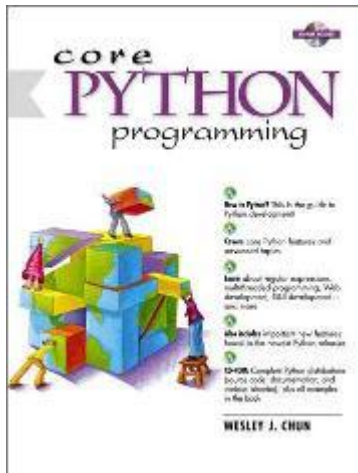
Advanced search

Core Python Programming

Michael Baxter

Issue #85, May 2001

Finally, a book good enough to be both a textbook and a reference on the Python language now exists.



- Author: Wesley J. Chun
- Publisher: Prentice Hall PTR
- URL: <http://www.phptr.com/>
- Price: \$44.99 US
- ISBN: 0-13-026036-3
- Reviewer: Michael Baxter

Finally, a book good enough to be both a textbook *and* a reference on the Python language now exists. Part of Prentice Hall's PTR Core Series, this book works well as a first Python textbook for computer science students, while also being thumbable for work-a-day use by Python programmers.

The book divides Python into two main parts. In Part I, about 500 pages explain the rudiments of Python: how to program in Python, the syntax, data types and key language features, all cleanly divided into 14 chapters. Each chapter is

compartmentalized yet is also incrementally linked to the others. Part II covers six different advanced topics in as many chapters in a little under 200 pages. In Part II, the advanced chapters accomplish what most Python books fail to do: show you integrated working examples of how the language is used to solve real-world problems.

For students, Part II allows you to apply directly what you learn from Part I, moving beyond just pushing what might initially be abstract symbols and syntax. For developers or programmers, Part II is a very nice HOWTO, which is strongly backed up by Part I as a language reference. For all users, the book invites exploration and satisfies your curiosity through modifying the copious examples. Some in-depth examination will illustrate these points.

To introduce the basics, Part I has chapter topics titled "Welcome to Python!", "Getting Started", "Syntax and Style", "Python Objects", "Numbers", "Sequences: Stings, Lists, and Tuples", "Dictionaries", "Conditionals and Loops", "Files and Input/Output", "Errors and Exceptions", "Functions", "Modules", "Classes and OOP" and "Execution Environment".

The order of topics in Part I really simplifies learning Python, especially if this is your first computer language. If you've ever had a programming class before and used a text that has exercises where you were forced to write code that did some silly language-specific action, you'll be pleased to know that this is *not* that kind of book. Plentiful, practical examples are used as each topic develops; the exercises are a lot more about exercising your creativity rather than some didactic language feature. Consider this one of the "fun" computer language books.

Because of the excellent depth presented in the introductory topics, the material is also of value to practicing Python programmers. For instance, most language books have tables and lists about features that are broadly discussed in the text. This book goes much further. The standard types in Python are explained compellingly in terms of storage model, update model and access model. These three simple concepts explain *all* of the possible interactions of data types in the language, doing more to help understanding than dozens of pages of prose. And, there are other examples of this kind of thought-provoking mastery of Python. A flowchart explains how Python does numerical coercion and is much easier to follow than any table or list. The best explanation ever achieved of how Python does slicing of sequence objects is shown using cartoons of soccer players. A lot of the power of Python comes from lists and dictionaries. Interestingly, explanations are given for what these objects do *not* do, as well as for their capabilities. These pages should be earmarked for reference use.

In Part II, we encounter the advanced topics. Again, this is where the book excels: it actually shows you *how* to do the programming for specific application areas involving “Regular Expressions”, “Network Programming”, “Multithreading Programming”, “GUI Programming with Tkinter”, “Web Programming” and lastly, “Extending Python” with alternative, compiled languages. In contrast to the interlinked development of Part I, the chapters in Part II are more individually self-supporting. They can be used as one has an interest or need for the information. The chapter on multithreading, in particular, is one of the only complete HOWTO developments on this topic.

Mention must be made of the valuable appendices and the index. Appendix A provides answers to selected exercises, for self-checking or class use. Appendix B contains one of the most comprehensive bibliographies of print and on-line references on Python. The operators in Python are conveniently collected in one place in Appendix C. The book keeps you completely up-to-date with Appendix D, which previews the new language features of Python 2.0. The index is impressively comprehensive at 17 pages in length.

The two-part division of the material feels very appropriate. For use as a textbook, the pacing and thoroughness are more than adequate. As a reference book, the clean division of the subject, plus a good index, allow rapid finding of any topic. A wide variety of examples are used in each chapter. This markedly increases the likelihood that something of salience matches your needs as a programmer. And while opinions vary, I tend to prefer books that offer a CD-ROM of multiplatform binaries and sources, as this one does.



Michael Baxter has been working in computer technology since he was nine, imprinted by a 1969 viewing of 2001: A Space Odyssey. He is an experienced computer architect, system, board and FPGA logic designer. Michael holds ten United States patents in computer architecture and logic, plus four patents as a coinventor. His Linux-related interests include open-source Verilog and EDA tools, Python, automated source code generation, concurrency and hardware issues.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

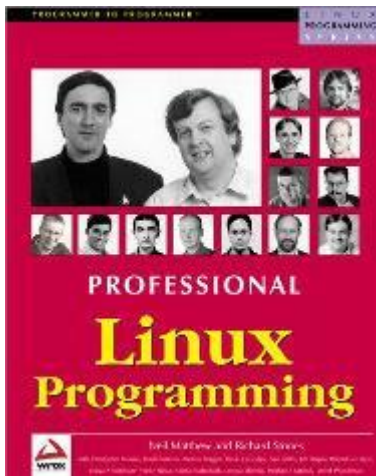
Advanced search

Professional Linux Programming

Stephanie Black

Issue #85, May 2001

This book is a huge credit to both its authors and publishers.



- Authors: Neil Matthew and Richard Stones
- URL: <http://www.wrox.com/>
- Price: \$59.99 US
- ISBN: 1-861003-01-3
- Reviewer: Stephanie Black

With the increasing number of Linux programming books on the market, the new Linux coder is left in a bit of a muddle.

Information about Linux is spreading—a good thing in terms of the open-source model and the community that supports it. The influx of new Linux authors also tends to decrease the odds of finding literacy and technical expertise in the same place. Numerous would-be writers ride the popularity wave of “Penguin Power!”, resulting in increasing numbers of Linux “bibles”, most of which are just as difficult to follow as the original and not nearly as imaginative. In either case, the context is most often distorted, if presented at all. Then again, there are exceptions.

Professional Linux Programming is a huge credit to both its authors and publishers. It's clearly written, highly informative and, with the exception of one or two topics, exhaustive in scope. Its intended audience is intermediate programmers, but the book is written in sufficiently clear and plain language as to be quite easily understood by the novice programmer.

The book's main "theme" is a project designed for a DVD rental store, taking into account both general and user requirements for a design specification. These specifications are followed up in subsequent chapters, which discuss various elements. The vast majority of the chapters in the book go back to this project and approach the subject matter from that context. Occasionally, the authors sprinkle in "take a break" chapters. These are related to a variety of programming issues in general. The project, however, is the meat of the book.

This approach provides not only a context, but a flow of information from one topic to the next. As a result, the reader learns a vast quantity of subjects easily because they're all connected to each other—an approach more conducive to increasing a coder's knowledge and skill than the "bits and pieces" approach so prevalent among technical works.

Highlights

The majority of programming books, even Linux programming books, are negligent about providing sane information regarding application design according to general (and user) specifications. Given that the first step of writing an application is to define its use and user-base, design is too often overlooked or, if mentioned, relegated to the back of the book. This is the *first* topic of discussion in *Professional Linux Programming* and gives the reader a taste of rationally approaching software development.

Having read several Linux programming books, all of which ignore (by intent or accident) the increasing popularity of CVS (concurrent version system), I was overjoyed to find an entire chapter detailing this marvelous tool, both in collaborative and individual contexts. Common CVS terminology (check-out, commit, repository, etc.) is delineated, as are commands and the growing number of CVS resources available on the Internet.

Databases are, for many programmers, the singularly least interesting subject and the one that is most easily overlooked. Matthew and Stones not only give a foundation in database requirements and functions, but offer an in-depth comparison of PostgreSQL and MySQL. An interesting addition was a brief contrast between libpq and ecpg as ways of calling PostgreSQL from C. Later in the book, as application construction confronts the area of connectivity, PHP is detailed as well.

In separate chapters, both debugging and testing are carefully illustrated, with particular attention paid to the varieties of software errors and the tools to intercept them prior to release. Testing tools, including different kinds of tests for different kinds of requirements, are examined in great length with particular attention paid to getting the most information from tests to ensure quality of the application.

Chapter 12 deals with secure programming (including a reference to Ken Thompson's "trojan" that appeared in every release of the UNIX compiler). This chapter does not, however, simply leave the reader with a simple alert; it details how permissions work, some useful cryptography tools and some environment variables, all of which can assist in the production of secure code. It ends with a discussion of security features/impediments that come with some of the more common programming languages.

I personally have to give an extra round of applause to anyone who can take the complexities of Beowulf clusters and turn them into very comprehensible English (I am currently scrounging a bunch of 386s with which to construct my own Beowulf cluster—or should that be Bayou-Wolf?).

Where the Book Succeeds

The success of the book lies in several factors, not the least of which is readability. This is a book that can be read cover-to-cover: for a tome of 1150+ pages, such quality is remarkable. The information is presented clearly and leaves the reader with a sense of having gone on a rather enlightening journey (without the flight delays).

Another factor in its success is the timeliness of its topics: this is information from which *all* programmers can benefit given current trends in software development. It takes into account the increasing need for networkable applications that don't require extensive training time for staff to become comfortable with them. The book also presents a vast array of less commonly discussed tasks such as LDAP (Lightweight Directory Access Protocol), building device drivers and distributing the application. Such topics may, in time, prove to be the direction of information technology. For now, they are useful as foundations on which to build increased repertoires of environments for which applications are needed.

There are two primary authors, but another eleven contribute information based on their individual areas of specialization: Browne, Clements (Python), Froggatt, Goodger (Python), Griffin, Licquia, van Loon (multimedia), Ranawake (Beowulf), Rawat (networks), Sundbakken (QT), Thomas (PHP), Turnbull (internationalization) and Woodhouse (device drivers). The synergy behind such a collective effort is apparent, as each contributor's work, experience and

expertise have created a comprehensiveness to *Professional Linux Programming* not often found in "tech-lit". If only more authors follow these gentlemen's fine example of collaboration.

The publishers, Wrox Press, even provide a support site for programmers: <http://p2p.wrox.com/>. This includes source code used in the book, as well as a forum for discussing programming issues.

Where the Book Falls Short

If you're a Debian developer, or want to be, this book will help in numerous respects, but there is no discussion of the .deb package format, or its relevant tools. Given the increasing numbers of developers who want to develop applications for Debian (or Corel, or Storm, or Libranet, or...), the topic would have merited at least a passing mention. The Connectiva Discovery (a version of APT that is RPM-aware) aside, the ubiquity of RPMs negatively impacts their wonderfulness.

Additionally, there are no autographs in the book. One can only assume this is rectifiable solely through writing to the authors and asking them sweetly for their missing signatures (you may need to send return postage).

Ladies and gentlemen, we have a winner!



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](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Letters to the Editor

Various

Issue #85, May 2001

Readers sound off.

Tech Topsy

The Tech Tip on page 16 [*Linux Journal*, March 2001] is great! Unfortunately, your tech ignored least common multiples and lowest common denominators. With the numbers given in paragraph five, you will **fsck** all four filesystems every 15th reboot, making the problem worse than the default scenario.

A better approach is to use prime numbers like 13, 17, 19 and 23. This way, you won't fsck two filesystems until the 221st reboot (13×17), and you won't fsck all four filesystems until the 96,577th reboot. Assuming an average uptime of 90 days (bad hardware, security updates), this is in the year 25,814.

—George Toft

Awkward Solution

I found Mick Bauer's proposed solution on page 33 ["Paranoid Penguin", March 2001] a bit awkward at best. Renaming a start script this way will result in a failure if you upgrade. (For example, **ude** to a security fix in the package.)

The preferred way for Red Hat Linux is to use **chkconfig**. So the sample should be:

```
chkconfig named off
```

For other distributions, you should move it to K70named (that is 100-n) or use whatever system that distribution uses.

—Hugo van der Kooij

Bauer replies: You are of course correct, chkconfig is the preferred way to manage startup-scripts in Red Hat. I didn't know this, having only recently switched from SuSE to Red Hat. But “awkward”? My way is common practice on most of the SysVInit implementations I deal with, including SuSE (unless SuSE 7.1 is different—haven't tried it yet). Regardless, I consider this a minor point: any upgrade “failure” caused by my method would be easy to fix. The only such weirdness I've experienced myself has been the occasional creation of redundant symbolic links, which I'd hardly categorize as a failure.

Mandrake 7.2 Review

I am greatly concerned about the review of Mandrake 7.2 found in the March 2001 issue. To cut to the chase, Mandrake works almost flawlessly out of the box, and many of the problems were as a result of the reviewer trying to install Helix GNOME on top of the Distribution (Helix GNOME has known errors and does not support Mandrake7.2). Taking screen shots in the GIMP does work out of the box, just not with the Helix version, which was obviously tacked on after the fact. The lack of a back button is absolutely false because the installer does not need a back button. (The installer screen has icons on the side which allow you to jump to any point in the install and shows you where you are in the installation.)

The sidebar says that Mandrake includes Helix GNOME; this is false. It does include GNOME, but not Helix. In fact, Helix does not even support Mandrake 7.2 (but it does support earlier versions).

I personally run Mandrake 7.2 both at work and at home, out of the box, and it runs almost flawlessly (though there are a few minor issues, but updates are available). I have also tried the GNOME formerly known as Helix on one of my 7.2 machines and found that it did break many things. The point is, Mandrake 7.2 worked fine out of the box and only started breaking after adding Helix GNOME!

—Chris Hendrickson

Black replies: As clearly stated in the article, the review I did was, evidently and provably, not of the final released version. I twice inquired of Mandrakesoft regarding this and they would not answer my e-mail. None of the Helix/Ximian problems evaporated with a clean, Helix/Ximian-free installation, and while the stars on the left side of the screen do work as back buttons, this sure isn't obvious to the Mandrake newbie. I have heard some wonderful things about the actual release version of Mandrake. I'm thinking about downloading it and putting it on a server, as it seems to work especially well in that capacity. Its security is well-noted. But as a reviewer, I can only work with what is sent. I can do the research, ask the questions (or try to), etc., but I can't say “well, gee, this

is marvelous” if I can't run the software in a normal fashion. I have a rule for developing software: if I can break software with my innocuous little system, then that software probably needs fixing anyway.

OpenSSH and DSA Keys

Mick Bauer's “The 101 Uses of OpenSSH: Part 2” [February 2001] is an excellent article with a small flaw. He writes: “To specify a particular key to use in either an ssh or scp session, use the -i [flag].” He also provides an example that suggests the use of DSA keys.

However, OpenSSH did not support the use of DSA keys with the -i flag until the very latest version (2.5.1, released just four days ago). Earlier versions silently ignored the DSA key indicated (RSA keys work just fine). Hence, anyone trying that example will see ssh mysteriously default to password authentication every time.

This limitation is actually documented in the ssh man page. Versions prior to 2.5.1 said:

```
-i identity_file<\n>  
  Selects the file from which the identity  
  (private key) for RSA authentication is read. [...]
```

Version 2.5.1, of course, says “...RSA or DSA...”. However, it's easy for even experienced users to miss the distinction—I certainly did the first few times.

—Adrian Ho

Missed the Tripwire

In the article “Designing and Using DMZ Networks ...” [March 2001] Mick Bauer covered ways of securing DMZ hosts. In my opinion he missed a simple but very efficient tool to detect intruders at DMZ hosts or firewalls: Tripwire. It calculates checksums for all files on the system and stores these fingerprints in a database. Doing a compare run against this database in regular intervals (cron), it is easy to detect changes. If somebody modified /bin/login, /etc/passwd or installed other back doors, you will realize it at least after the next compare run. The key is to store Tripwire itself and the initial database on read-only media (e.g., CD-ROM) to prevent modifications. There is no way in doing the same with **diff**, as mentioned in the article.

Tripwire is commercial software now, but there is a GNU GPL edition for Linux available at www.tripwire.org and www.tripwire.com/. There is another GPL'd software dealing with the same subject, but I never tried it. It is at www.cs.tut.fi/~rammer/aide.html.

Another issue is the design of the DMZ shown in Figure 2. I wouldn't recommend having all hosts in a single DMZ. If you are using three different boxes for doing the job, you should use three DMZs as well. If one of the machines is compromised by an intruder, he has to cross the firewall again to attack the others. So fill up your firewall with additional NICs and use crossed cables—you won't need a switch either.

Regards

—Markus Hogger

FAT Problems

I enjoyed Robin Rowes' article, "Debian Multiboot Installation" *LJ*, March 2001, but have a couple of points to make about it. In the part about running `rawrite2.exe`, it is implied that you can't run this program from a FAT32 partition. This is wrong; the versions of DOS that come with Win98 (and Win95 OSR2) know about FAT32 (but not long filenames); otherwise, they wouldn't be able to boot Windows from a FAT32 partition either. Perhaps the author had FIPS (the DOS repartitioning tool) in mind at the time, the original versions of which cannot handle FAT32.

A discussion of the problems with WindowsME would have been useful. This is basically Windows98 with a flashier GUI and other useless features, except Microsoft tried as hard as possible to stop you from running its DOS in "real" (16-bit) mode, mainly by nobbling the `FORMAT` and `SYS` commands and removing the options for starting or restarting in DOS mode.

To boot to DOS in WindowsME, create a startup floppy from Control Panel --> Add/Remove Programs --> Startup Disk. If you reboot the PC from this floppy and select the Minimal Boot option, you will end up at a DOS prompt from which you can change to drive C: and run the `rawrite2.exe` program as instructed in the article. Alternatively, you could get your own back on Microsoft by nobbling the startup floppy to get CD support and a DOS prompt without any of that Windows recovery malarkey. (The easiest way is just to rename the `AUTOEXEC.BAT` file.)

Also, a lot of the pathnames in the article use forwardslashes instead of backslashes.

—Ian Abbott

Rowe replies: You are correct that covering WinME would have been nice. The only reason I didn't was I don't have a copy and didn't want to buy one. Thanks for the nice notes on how to use it. Another interesting approach that I haven't

tried is using WinImage to create the Debian floppies. I think you are right about being able to see a FAT32 partition when booting from a newer DOS. I haven't tried that in a long time, since I generally prefer FAT16 or NTFS partitions. I should have created a FAT32 partition to test that, but was in a rush to complete the article and didn't. Thanks for the correction. I've never used FIPS. Forwardslashes are correct in UNIX or Windows, although Windows persists in defaulting to backslashes. Using forwardslashes everywhere is a habit I picked up in writing portable code. Unfortunately, that only works with UNIX and Windows. The Mac doesn't like slashes (forward or back). Darwin will, I hope, change that. If you go to the file search box in Win2k, for instance, and use forwardslashes, that works fine. The one place it won't accept forwardslashes is at the DOS command prompt.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

UpFRONT

Doc Searls

Issue #85, May 2001

Stop the Presses, *LJ* Index and more.

Public Execution

You arrive at the airport, in a hurry as usual. You look up at the flight departure screen and it says “A fatal exception OE has occurred at....The current application will be terminated...” You know the rest. This is the epitaph in blue that Windows issues for every application it terminates, also known as the Blue Screen of Death, or BSOD.

Lately Matt Michie, a writer and Linux hacker, has been keeping a growing gallery of public BSODs at his site (<http://www.daimyo.org/bsod/>), which ranks high among the many thousands of pages that show up in searches for “BSOD” or “Blue Screen of Death”. From what we can tell, these include a giant Las Vegas billboard, an airport flight schedule display, an ATM machine, a casino game, a building marquee and public signage of various unknown, but not unembarrassing, sorts.



LJ Index—May 2001

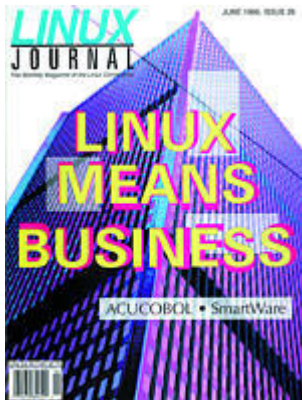


1. Number of e-mail packages other than Microsoft Outlook Express that could be infected by the Anna Kournikova virus: 0
2. Number of e-mails carrying the Anna Kournikova virus received by one *Linux Journal* editor on February 11, 2001: 15
3. Approximate number of people laid off by Eazel in March 2001: 40
4. Approximate number of SuSE employees at the time, worldwide: 600
5. Number of French schools now involved in the national Linux program: 350
6. Number of Linux distributions: 188
7. Number of embedded Linux OSes: 25
8. Total number of embedded Linux OSes as of last August: 17
9. Percentage increase in thin-client shipments during 1999: 90
10. Percentage of *Fortune* 100 companies that use thin clients: 80
11. Size of Amazon.com's announced layoffs in January 2001: 1,300
12. Size in millions of dollars of Amazon's stock trust fund for laid-off employees in January 2001: 2.5
13. Percentage of internet usage within the US population in February 2001: 60
14. Millions of people logging on to the Internet in the US in February 2001: 169
15. Work access as a percentage of all net access in the US in February 2001: 14
16. Billions of wireless messages sent per month by the end of 2004: 244
17. Number of MP3s in the average peer-to-peer user's hard drive: 500

Sources:

- 1: News.com
- 2: Doc Searls
- 3-4: Michael Hasenstein
- 5: LinuxWorld

- 6-8: TechWeb
- 9: International Data Corp.
- 10: Giga Information Group
- 11-12: Moneybox
- 13-15: News.com
- 16: News.com
- 17: Eric Garland of BigChampagne.com, from *The Standard*



LJ History

In the June 1996 issue of *Linux Journal*, we wrote that in addition to support for IA32, Amiga, and Atari, Linux is “being ported to several more platforms, including Alpha, ARM, MIPS and PowerPC.” Five years later, not only are these supported but also SuperH, SPARC and UltraSPARC, the S/390, VAX, PA-RISC, IA64, ColdFire, DragonBall, ETRAX, i960 and a host of others.

Microsoft and PHP gain in Netcraft survey

At this point it almost goes without saying that the predominant web server is Apache, which has had the top market share across all domains since early 1996, holding at better than 60% for the last year—according to Netcraft who has been studying the matter since September 1995. Back then the top server by a huge margin was NCSA's, which flatlined at approximately zero in 1999.

Apache is still doing fine, but it dropped by 1.29% in January of this year, Netcraft reports. Microsoft's IIS gained 1.82%, and Sun's iPlanet, the third-place server, dropped .29%. Both Apache, and Microsoft gained in absolute numbers. Netcraft found 16,207,982 domains served by Apache and 5,903,512 served by Microsoft. But there were other developments behind those details. Netcraft reports,

The relatively static market share for Microsoft on the Web as a whole contrasts sharply with its progress in our companion SSL Server survey where Microsoft makes consistent and relentless gains, month after

month, and now accounts for 49% of the sites performing encrypted transactions on the Internet. Arguably, Microsoft's applications have made the difference, with there being no straightforward alternative to Microsoft's Commerce Server in the UNIX world.

Netcraft also reveals that PHP is being adopted at an extremely rapid pace:

The PHP module is compiled into the Apache server on over five million web sites, or approaching 40% of all Apache sites. Although PHP will only be in use on a fraction of these sites currently, it is regarded as easier to program than Perl or JSP and has created a broad developer community in a relatively short space of time. PHP, together with MySQL and Apache, has become the de facto way of developing web applications in the Linux environment, in a similar way to the IIS/ASP/SQL-server combination in the Microsoft world.

The Netcraft survey can be found at www.netcraft.com.

Stop the Presses: a Love Letter from Big Tux to Big Bux

None of the Big Boys have gone as gaga over Big Tux as Big Blue. We all know about the billion IBM plans to spend. We've heard about the server division changes and the R&D commitments. We've seen the beanbag chairs rolled out for geeks at LinuxWorld Expo.

But, that's not enough. We're talking about a huge crush here: the kind that calls for big sloppy displays of unfettered affection. But since large corporations aren't able to actually hug and kiss (it would never get past Legal), IBM can't help doing what comes only naturally to big old companies—advertise.

“Peace, Love and Linux” is the new campaign. It launches with a love letter in the form of a Flash Q&A on the Web. Here's how it starts:

What is “Peace, Love and Linux?”

A rallying cry. A clear, enthusiastic synopsis of IBM's excitement about Linux and support for the Linux community.

Why is IBM supporting Linux?

Because we admire it, we believe in it, we need it and it's good for customers.

IBM's vision is to help build the business infrastructure of the future, drive towards dynamic e-business and

integrated, intelligent infrastructure. The complexity and demands of this vision are staggering, mind-boggling. We've looked hard at what it will take to get there, and two facts have become clear.

First, we'll never get there without wide adoption of open standards like Linux.

Second, that the complexity of the task is so great that IBM simply can't do it alone. No company can do it alone (although some still suggest otherwise). Only the concerted effort of the larger technology community can make it happen. And only the Linux movement can marshal that effort.

Here's what IBM says to the natural skeptics that comprise, frankly, our readership:

Is this collaboration idea real or just feel-good stuff?

Real. Very real. If you think it's just fluff, you've missed the point.

IBM spends \$5 billion a year on R&D. And we're putting a billion dollars behind Linux. But even all that is nothing compared to what the Linux community will generate spontaneously.

This is a new way of looking at the world: companies, alliances, partnerships and individuals working together, each contributing their particular expertise to create a "massively parallel thinking" force that is immeasurably more powerful than any single entity, including IBM.

As always with this kind of stuff, the lover holding out the flowers makes a few little mistakes. For example, IBM doesn't appear to know about the Linux community's allergy to GIFs. To arrive at the love letter (www-1.ibm.com/servers/eserver/linux/passport.swf), readers of IBM's Linux page have to click through a series of GIF links. And, lots of us will be disappointed when we get there because they don't run Flash, either.

So a little advice to the suitor here: maybe you could take some of the web marketing money that's burning a hole in your pocket and bang some of it on, say, the Open Source Initiative (<http://www.opensource.org/>). They need it, and it'll help them accomplish a bunch of the objectives you're bragging about.

—Doc Searls

They Said It

We've just spent 14 hours trying to get standards-compliant code to work in standards-compliant browsers.

—Jeffrey Zeldman

The Linux issue is whether this is a fundamentally disruptive technology, like the microprocessor and the Internet. We're betting that it is.

—Irving Wladawsky-Berger of IBM

It's the business model around free software that concerns us, where people get sucked into not paying for software. This will be a disservice for them, as they need established, well-funded companies to offer innovation.

—Doug Miller, group product manager for Microsoft's Windows Server Group

The Linux community support model has resonated with people.

—Steve Ballmer, Microsoft CEO

In this business, the only real open-industry standard in the computer industry is Linux, which thankfully remains beyond the clutches of the moguls. Everything else is hokum designed to lock developers (and by extension, customers) into proprietary corners of the computing constellation.

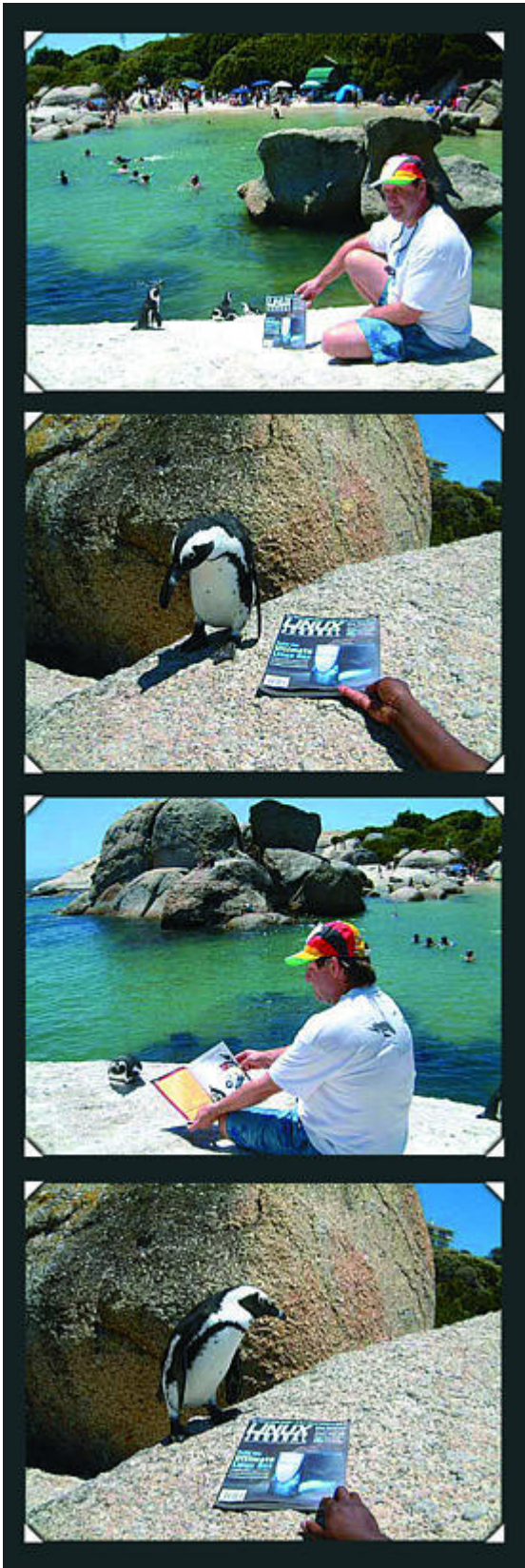
—Charles Cooper, ZDNet News

Life ain't a rehearsal; stop acting and live.

—EliahuOne

Copyright today is a system inflicted on the public, not a system that benefits the public.

—Richard Stallman



[Archive Index Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

Best of Technical Support

Various

Issue #85, May 2001

Our experts answer your technical questions.

PHP Can't Connect to MySQL

I installed MySQL3 and PHP4 with an RPM package. Apache is running well and the MySQL server is running well from the shell. PHP4 also runs with the `infophp()` function call. But I can't connect to the MySQL server. What's wrong? —Haidar AM, haidar@netexecutive.com

You did not mention it, but I am assuming that the `phpinfo` script showed that you do have support for MySQL. If this is the case, please check the MySQL table to make sure that the user/password/host, localhost in this case, are correct. —Mario Neto, mneto@argo.com.br

Automating Keyboard Tasks

I've been handed the task of automating various software tasks for a company and a lot of it will entail emulating keyboard input. How can I echo key commands from a script to a program? I've not even been able to find a thorough list of what keys map to what escape sequences. Can anyone help? —Paul, tanx1@stlnet.com

Try the **expect** command. *Linux Journal* published an article about it a couple of months ago [That would be the December 2000 issue, Mario—Ed.]. It comes with most distributions and lets you write scripts so you can automate interactive sessions. —Mario Neto, mneto@argo.com.br

One simple method I use to call some programs from a script is **echo "y" | xyz**. In this case I run `xyz`, which will, after starting, get a "y" in response to its question. —Usman S. Ansari, uansari@yahoo.com

Can't Log in as Root

For some reason I cannot log in as root at my local Linux box (not remote login). I can log in as any user and then su to root. Second, whenever I have to execute any commands like **adduser**, I have to specify the absolute path as to where the command lies such as (/usr/sbin/adduser). —Devraj Sen, devraj@trihedral.com

The file /etc/securetty is used as a list of terminals from which root is allowed to log in. So, it is possible that the entries for your console logins were removed. Simply put them back in, i.e., to allow root to log in on the first console you should have a line consisting of:

```
tty1
```

The reason why your PATH isn't set up properly when you use su is because it isn't being treated as a login shell. So, none of the login files that pertain to root are being read. To have them read, type **su -** instead. —Andy Bradford, andyb@calderasystems.com

As far as I can tell, most if not all, Red Hat releases have been broken in so far as /sbin:/usr/sbin:/usr/local/sbin has never been in root's path for various reasons. The quick way to fix that is to simply add it to the PATH in /etc/profile. As you cannot log in as root, this is not usual behavior. You should make sure root has a valid shell, and that /etc/securetty still contains "ttyx" lines. For more clues, type **tail -f /var/log/messages** and look at the output when you try to log in as root from another console. —Marc Merlin, marc_bts@valinux.com

Updating the Shadow File

What is the command for synchronizing/updating the shadow file after I create a user account by manually editing the password file? Also, how do I remove the suid bit from any process? —Kedar, kedar.potdar@imandiasia.com

pwconv is the command for synchronizing/updating the shadow file after creating a user account. —Usman S. Ansari, uansari@yahoo.com

To remove the suid bit use **chmod -s filename**. —Marc Merlin, marc_bts@valinux.com

To protect against accidentally mangling /etc/passwd if someone runs a utility that touches the file at the same time you're editing it, use **vipw** any time you edit /etc/passwd (it uses your chosen editor, not necessarily **vi**). —Don Marti, info@linuxjournal.com

Perl Delays Writes, Then Dies

I have written a basic Perl program that reads a list of URLs from a file, goes to the URL, looks for some information and then writes that information to another file. It also writes entries to a log file and stdout. I am using the LWP::Simple module. I note the following strange behavior: 1) the log file is not written to immediately—the OS seems to be caching write requests; and 2) the program seems to die after 1-2 hours of perfectly normal operation, but restarting it works fine. —Dave Barter, dave@phased.co.uk

Typing `$| = 1` at the beginning of your script will force your Perl program to do a flush after every print or write (see `man perlvar`). As for why your program is dying, try running `top` at the same time to see if it's hogging all your memory. If so, rewrite it to forget about pages after it's done with them. If you run your program unattended, you should be using LWP::RobotUA, to respect webmasters' wishes about what parts of their sites are open to robots. —Don Marti, info@linuxjournal.com

Corrupted Root Filesystem

I am trying to install Red Hat onto my Dell laptop using the partition I created for it. The install runs successfully, but when I try to load Linux it hangs up after these messages:

```
checking root filesystem
/ was not cleanly unmounted, check forced
Setting filetype for entry log in /dev (174593) to 6
Unattached inode 82384
UNEXPECTED INCONSISTENCY; run fsck MANUALLY
(ie without -a or -p options)
An error occurred during the filesystem check
Dropping you to a shell; the system will reboot
Give root password for maintenance
```

—Brian Weigner, brian.weigner@colorado.edu

Seems your Linux root filesystem was corrupted somehow. You actually need to get into single-user mode by issuing a `linux s` command at the LILO prompt and wait until you get the `#` root prompt. Type:

```
e2fsck -r /dev/your-root-disk-device
```

This will go through the device and ask you what to do with each error it finds. This will probably ask many questions that should be answered. To properly answer all of them, some Linux filesystem experience is a must. This is risky business on a real production system. For now, since it is a fresh install and to make it easy, answer yes to all the `FIX?`, and `REPAIR?` and `REMOVE unused/dirty stuff`, questions. After you are done, try rebooting your system. If that doesn't

work, maybe your best bet is to carefully reinstall. —Felipe E. Barousse Boué, fbarousse@piensa.com

To prevent this from happening in the future, run **shutdown -h now** (as root) before you power down. —Don Marti, info@linuxjournal.com

FTP Users Go Ape! Virtual Hosts Fail!

My FTP users are able to back past their home directory and go right to /. They are using leach FTP, and it allows them to go up levels even in their home directory.

Also, I have 20 virtual sites (by name, not IP) running on my Apache server. A couple of times when adding a new virtual host, it would not resolve until I moved it to the top of the list of virtual hosts in httpd.conf. Am I out of virtual servers? The computer is an AMD K6 200MHz with 64MB of RAM. —Aaron, aaron@x56.net

There is a hack in **wuftp** to prevent users from cd-ing up, but you are probably better off installing proftpd if you aren't using that yet. It supports chrooting users in their home directory or some other preset directory. See: <http://www.proftpd.net/>. —Marc Merlin, marc_bts@valinux.com

You're not out of virtual hosts, but this configuration can be tricky. Here are some things to check. Do any of the ServerName or ServerAlias values in the virtual hosts before the broken one match the broken one's name? If you have a ServerAlias *.example.com and add warez.example.com, it won't work. Keep everything with a * alias at the end. Do an **nslookup** to make sure that the DNS for all the virtual hosts works. Make sure that the NameVirtualHost directive is before the corresponding VirtualHost sections. Read <http://docs-2.0/vhosts/details.html> and, of course, every webmaster's most-read and least-enjoyed reading material, error_log. —Don Marti, info@linuxjournal.com

I Have No Dialtone, and I Must Dial

I have a PCTEL internal modem, and I got the drivers from linmodems.org and have installed them. The problem is the modem keeps on dialing with the message NO DIALTONE. I was able to connect only once and that was after trying for a long time. Also, the transfer rate wasn't what I was expecting. —Krishna, as_krishna@hotmail.com

Try to see if adding X1 in your AT init string makes a difference. —Marc Merlin, marc_bts@valinux.com

Spammers Fishing for Addresses

Increasingly, I see spammers attempting to send mail to my domains by simply trying a series of common first names (i.e., david@example.com, bill@example.com, mike@example.com, etc.) Is there any automated method of shutting down the connection after X failed addresses and, even better, adding their IP to sendmail's access database? —Waldo Jaquith, waldo@waldo.net

You could write a small Perl script that parses the reject log, adds the IP to a blacklist and restarts sendmail when this happens. —Marc Merlin, marc_bts@valinux.com

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

New Products

Heather Mead

Issue #85, May 2001

48-Port Network Access Card from Ariel, RackMount-1UAXe from Rave Systems and more.

48-Port Network Access Card from Ariel

A 48-port version of Ariel's RS4200 high-density 56K/ISDN PCI network access two-card set is now available. The RS4200 combines T1/PRI or E1/PRI interfaces with 56K and basic rate ISDN remote access ports to build a full-featured system with remote dial-in and LAN dial-out. The 48-port card set can be field upgraded in 24-port increments to handle up to 120 ports. A Java-based remote management tool comes with the card set so that administrators can monitor an unlimited number of RS4200 ports located on the LAN or Internet. Other features include the ability to configure the T1/E1/PRI line interface and take resources out of service.

Contact: Ariel Corporation, 2540 Route 130, Cranbury, New Jersey 08512, 609-860-2900, info@ariel.com, <http://www.ariel.com/>.



RS4200

RackMount-1UAXe from Rave Systems

The RackMount-1UAXe is a thin server in a 1U form factor, powered by Sun's UltraAXe motherboard. Standard features are one 33MHz/32-bit PCI slot, two

front-accessible disk drives, an Ethernet 10BaseT/100BaseT port, 1GB maximum memory, on-board PCI graphics accelerator, one parallel port, one PS/2 port and four serial ports, 150-watt power supply and choice of Solaris or Linux. As an option, a third-party SCSI controller card can be configured for an increased data-transfer rate and larger disk capacity. The 1UAXe ships with a 300MHz UltraSPARC-IIi, 48x EIDE CD-ROM and Red Hat pre-installed.

Contact: Rave Computer Association, Inc., 36960 Metro Court, Sterling Heights, Michigan 48312, 800-966-7283 (toll-free), <http://www.rave.net/>.

USB 4-Port Serial Adapter from Keyspan

With this Keyspan adapter, four serial devices can be connected to a single USB port. Each DB9 port allows connection to RS-232 serial devices at data rates up to 960Kbps. This USB adapter supports Linux 2.4. Geared toward enterprise and industrial markets, it is useful for laboratory, manufacturing and retail setting for POS, process control or data retrieval applications.

Contact: Keyspan, 3095 Richmond Parkway #207, Richmond, California 94806, 510-222-0131, info@keyspan.com, <http://www.keyspan.com/>.



USB Adapter

BlueCat Linux for Intel XScale

LynuxWorks, Inc. announced the availability of BlueCat Linux for Intel's XScale microarchitecture, their new processor core technology. XScale is designed to provide high performance, low-power consumption and reduced thermal constraints for various market segments, such as wireless handhelds and internet infrastructure applications. An evaluation copy of BlueCat will be included with Intel's shipments of the IQ80310 XScale evaluation platform. BlueCat and XScale are geared toward storage, networking and handheld internet-capable products.

Contact: LynuxWorks, Inc., 2239 Samaritan Drive, San Jose, California 95124, 408-879-3900, <http://www.lynuxworks.com/>.

TinyTERM v4.13

TinyTERM is software used to provide secure client-based access to UNIX and IBM hosts over a company's intranet. TinyTERM also provides desktop PC users with networking utilities, including printer sharing, transparent printing, file sharing and drag-and-drop file copying. Version 4.13 gives users more control over where print jobs go and how they look. Features include a form feed for direct-to-device printing, improved buffering for large print jobs, a selection box that allows the printer to choose fonts, improved Telnet error handling and updated help files. TinyTERM v4.13 is available for download at <http://tt413.centurysoftware.com/> and is free for v4.12 users.

Contact: Century Software, 5284 South Commerce Drive, Suite C-134, Salt Lake City, Utah 84107, 801-268-3088, sales@centurysoftware.com, <http://www.centurysoftware.com/>.

2-Port RocketPort Serial Hub

Comtrol Corporation introduced a 2-port version of their RocketPort Serial Hub designed for industrial use. The DIN rail-mountable 2-port model minimizes cabling requirements by using existing Ethernet networks, eliminates redundant PCs and offers a backup port. The 2-port design enables support for faster Ethernet connections, more serial interfaces, real-time operating systems and connection to COM ports for simplified device connectivity. Comtrol provides support for 10/100Base-T Ethernet and connections with RS-232/422/485 and ModBus devices.

Contact: Comtrol Corporation, 6655 Wedgwood Road, Suite 120, Minneapolis, Minnesota 55311-3646, 800-926-6876 (toll-free), info@comtrol.com, <http://www.comtrol.com/>.



Front view of rocketport

KDE 2.1

KDE 2.1, an internet-enabled desktop for Linux, has been released by the KDE Project and is available for free download at <http://ftp.kde.org/stable/2.1/distribution/tar/generic/src/>. Features of KDE 2.1 include the Konqueror web browser, file manager and document viewer; the KDevelop IDE for

development; a network transparent multimedia architecture based on the Analog Realtime Synthesizer; the re-engineered KWin window manager; and improved stability, performance and feature set. KDE is available in 33 languages and ships with the core KDE libraries, desktop environment, developer packages and over 100 applications.

Contact: KDE Project, Kurt Granroth, 1456 S. Boulder Street, Unit D, Gilbert, Arizona 85296, 480-732-1752, granroth@kde.org, <http://www.kde.org/>.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.