Linux Journal Issue #91/November 2001

Open-Source Web Servers: Performance on a Carrier-Class Linux Platform

Ibrahim F. Haddad

Issue #91, November 2001

Ibrahim tests the performance of three open-source webservers on a typical Ericsson Research Linux clusterplatform.

ARIES (Advanced Research on Internet E-Servers) is a project that started at Ericsson Research Canada in January 2000. It aimed at finding and prototyping the necessary technology to prove the feasibility of a clustered internet server that demonstrates telecom-grade characteristics using Linux and open-source software as the base technology.

The telecom-grade requirements for clustered internet Linux servers are very strict and well recognized within the telecommunications industry. These characteristics include a combination of guaranteed availability (guaranteed 24/7 access), guaranteed response time (statistically guaranteed delays), guaranteed scalability (large-scale linear scalability) and guaranteed performance (to serve a minimum number of transactions per second).

In addition, telecom-grade internet servers have other important requirements to meet, such as the capability to cope with the explosive growth of internet traffic (growing at over 100% every six months) as well as meeting the increased quality of service demanded by the end users, not to mention very strict security levels.

These internet servers necessitate a high-performance and highly scalable web server. Since all of the work in ARIES is based on open-source software, we needed an open-source web server that could help us build our targeted system.

One of our goals in ARIES is to be able to build an internet server capable of scaling to thousands of concurrent users without download speeds noticeably

slowing. This type of scalability is best accomplished when application servers are hosted on a group or cluster of servers. When a request for a particular page of a web site comes in, that request is routed to the least busy server (using a smart and efficient traffic distribution solution, either hardware- or software-based).

We decided to experiment with three web servers: Apache, Jigsaw and Tomcat. Apache is the world's most popular web server. We have been experimenting with it since ARIES first started in 2000. Jigsaw, a Java-based web server, is currently used on our experimental Linux cluster platform. Tomcat, another Java-based web server, is a potential replacement to Jigsaw if proven to be a better performer.

The Apache web server is a powerful, flexible, HTTP/1.1-compliant web server. According to Netcraft Web Servers' survey, Apache has been the most popular web server on the Internet since April 1996. This comes as no surprise because of its many characteristics, such as the ability to run on various platforms, its reliability, robustness, configurability and the fact that it provides full source code with an unrestrictive license. For our tests, we have experimented with Apache 1.3.14, which was the stable release at the time, and the Apache 2.08 alpha release (2.08a).

Jigsaw is W3C's open-source project that started in May 1996. It is a web server platform that provides a sample HTTP 1.1 implementation and a variety of other features on top of an advanced architecture implemented in Java. Jigsaw was designed to be a technology demonstration to experiment new technologies rather than a full-fledged release. For our tests, we used Jigsaw 2.0.1 (serving HTTP requests on port 8001) in conjunction with the Java 2 SDK.

Tomcat is the reference implementation for the Java Servlet 2.2 and JavaServer Pages 1.1 technologies. Tomcat, developed under the Apache license, is a servlet container, a runtime shell that manages and invokes servlets on behalf of users, with a JSP environment.

Tomcat can be used either as a standalone server or as an add-on to an existing web server such as Apache. For our testing, we installed Tomcat 3.1 as a standalone server, servicing requests on port 8080.

Linux Cluster Configuration

For the purpose of testing and evaluating the above-mentioned web servers, we set up a typical Ericsson Research Linux cluster platform (see Figure 1).

Figure 1. Ericsson Research Typical Linux Cluster

This platform is targeted for carrier-class server applications. The testing environment consisted of:

- Eight diskless Pentium III CompactPCI CPU cards running at 500MHz and powered with 512MB of RAM. The CPUs have two onboard Ethernet ports and are paired with a four-port ZNYX Ethernet card providing a high level of network availability.
- Eight CPUs with the same configuration as the others except that each of these CPUs has a disk bank. The disk bank consists of three 18GB SCSI disks configured with RAID 1 and RAID 5 to provide high data availability.
- Master Nodes: two of the CPUs (with disks) act as redundant NFS, NTP, DHCP and TFTP servers for the other CPUs. The code for NFS redundancy was developed internally along with a special mount program to allow the mounting of two NFS servers at the same mounting point.

When we start the CPUs, they boot from LAN (either LAN 1 or LAN 2 for higher availability in case either of the LANs go down). Then they broadcast a DHCP request to all addresses on the network. The master nodes will reply with a DHCP offer and will send the CPUs the information they need to configure network settings such as the IP addresses (one for each interface: eth0, eth1, znb0 and znb1), gateway, netmask, domain name, the IP addresses of the boot servers and the name of the boot file.

The diskless 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 the DHCP server. Next, the CPUs will download a RAM disk and start the application servers, which are the Apache, Jigsaw and Tomcat web

servers. The process of booting a diskless server takes less than one minute from the time it is booted until we get the login prompt.

As for the CPUs with disks, they will download and boot the specified boot file in the DHCP configuration file, which is a kernel image located under the /tftpboot directory on the DHCP server. Next, they will perform an automatic RAID setup and a customized install for Red Hat 6.2. When the CPUs are up, they will start Apache, Jigsaw and Tomcat web servers, each on a different port. The process of booting a disk server takes around five minutes from the time it is booted until we get the login prompt (which includes an automatic RAID 1 and RAID 5 setup, as well as a complete install from scratch for Red Hat 6.2).

For our testing, we were booting the disk CPUs (six of them, except the master nodes) as diskless CPUs so we could have an identical setup on many CPUs.

The Benchmarking Environment

The performance of web servers and client-server systems depends on many factors: the client platform, client software, server platform, server software, network and network protocols. Most of the performance analysis of the Web has concentrated on two main issues: the overall network performance and the performance of web server software and platforms.

Our benchmarks consist of a mechanism to generate a controlled stream of web requests with standard metrics to report results. We used 16 Intel Celeron 500MHz 1U rackmount units (see Figure 2) that come with 512MB of RAM and run Windows NT. These machines generate traffic using WebBench, a Freeware tool available from www.zdnet.com.

Figure 2. Benchmarking Units

The basic benchmark scenario is a set of client programs (loaf generators) that emit a stream of web requests and measure the system response. The stream of requests is called the workload. WebBench provides a way to measure the performance of web servers. It consists of one controller and many clients (see Figure 3). The controller provides means to set up, start, stop and monitor the WebBench tests. It is also responsible for gathering and analyzing the data reported from the clients.

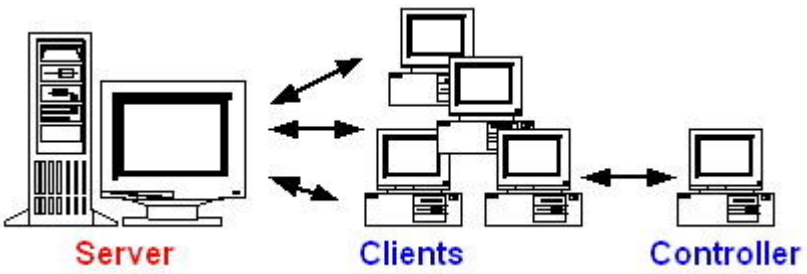


Figure 3. Architecture of WebBench

On the other hand, the clients execute the WebBench tests and send requests to the server. WebBench uses the client PCs to simulate web browsers. However, unlike actual browsers, the clients do not display the files that the server sends in response to their requests. Instead, when a client receives a response from the server, it records the information associated with the response and then immediately sends another request to the server.

There are several measurements of web servers. For our testing, we will report the number of connections or requests served per second and throughput, the number of served bytes per second (see Figure 4).

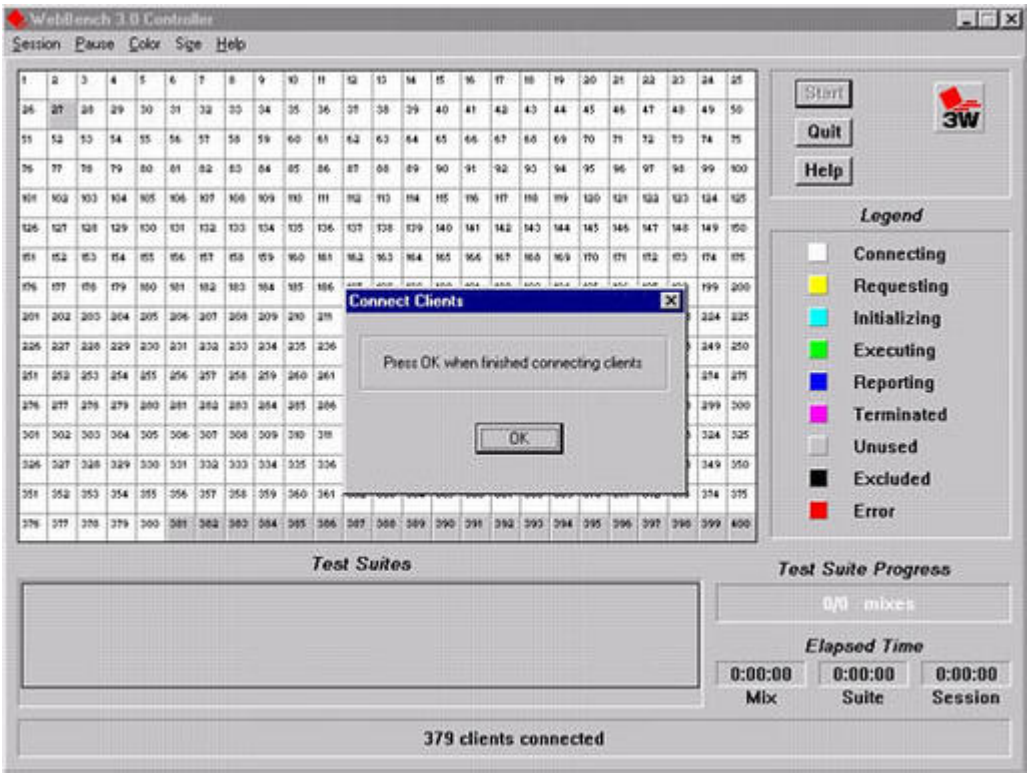


Figure 4. WebBench Control Window

WebBench uses a standard workload tree to benchmark the server. The workload tree comes as a compressed file that we need to move to the server and expand in the HTML document root on the web server (this is where the web server looks for its HTML files). This will create a directory called WBTREE that contains 61MB of web documents that will be requested by the WebBench clients. Since some of our CPUs are diskless, we installed the workload tree on the NFS server and modified the web server configuration to use the NFS directory as its document root.

As part of WebBench configuration, we specified that the traffic generated by the benchmarking machines would be distributed equally among the targeted CPUs. Figure 5 shows how we specify each server node and the percentage of the traffic it will receive.

Generating Traffic to 4 CPUs

CPU	%
CPU 1	25%
CPU 2	25%
CPU 3	25%
CPU 4	25%

Figure 5. Sample WebBench Configuration

Test Cases

After setting our Linux cluster and the benchmarking environment, we were ready to define our test cases. We tested all of the three web servers (Apache both 1.3.14 and 2.08a, Tomcat 3.1 and Jigsaw 2.0.1) running on 1, 2, 4, 6, 8, 10 and 12 CPUs. For every test case, we would specify in the RAM disk loaded by the CPUs which web server to start when the RAM disk is loaded. As a result, we ran four types of tests, each with a different server and on multiple CPUs.

For the purpose of this article, we will only show three comparison cases: Apache 1.3.14 vs. Apache 2.08a on one CPU, Apache 1.3.14 vs. Apache 2.08a on eight CPUs and Jigsaw 2.0.1 vs. Tomcat 3.1 on one CPU.

The first benchmark we did was to test all the web servers on one CPU. In WebBench configuration, we specified that all the traffic generated by all the clients be directed to one CPU. Figure 6 shows the results of the benchmark for up to 64 simultaneous clients. On average, Apache 1.3.14 was able to serve 828 requests per second vs. 846 requests per second serviced by Apache 2.08a. The latest showed a performance improvement of 2.1%.

Num. of Clients	Apache 1.3.14	Apache 2.08a
1_client	182.517	187.333
4_clients	731.067	735.783
8_clients	971.900	978.800
12_clients	931.500	946.067
16_clients	922.767	939.133
20_clients	933.217	943.333
24_clients	913.383	939.050
28_clients	903.333	935.717
32_clients	899.467	945.817
36_clients	867.700	894.633
40_clients	885.933	887.933
44_clients	782.283	828.650
48_clients	759.583	816.017
52_clients	817.150	821.700
56_clients	840.217	838.950
60_clients	856.833	864.133
64_clients	883.067	890.366

Figure 6. Apache 1.3.14/2.08a Benchmarking Data on One CPU

Figure 7 plots the results of the benchmarks of Apache 1.3.14 and Apache 2.08a. As we can see, both servers have almost identical performance.

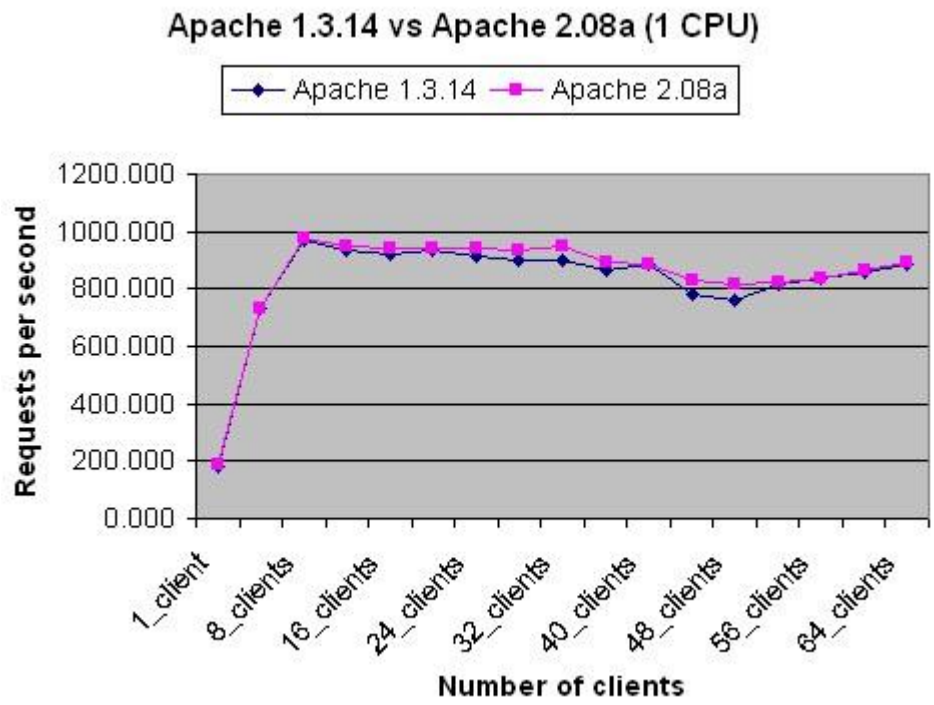


Figure 7. Apache 1.3.14/Apache 2.08a Benchmarking Results on One CPU

As for the Java-based web servers, Tomcat and Jigsaw, Figures 8 and 9 show the resulting benchmarking data. The maximum number of requests per second Jigsaw was able to achieve was 39 vs. 60 for Tomcat. We were surprised by Jigsaw's performance; however, we need to remember that Jigsaw was designed to experiment new technologies rather than as a high-performance web server for industrial deployment.

Nm. of Clients	*Tomcat*	*Jigsaw*
1_client	60.45	14.067
4_clients	68.283	24.296
8_clients	59.283	20.604
12_clients	54.267	18.304
16_clients	60.917	16.908
20_clients	57.067	16.721
24_clients	53.633	15.696
28_clients	56.2	19.533
32_clients	50.866	16.634
36_clients	50.784	16.183
40_clients	53.767	39.358
44_clients	58.816	36.583

Figure 8. Tomcat/Jigsaw Benchmarking Data on One CPU

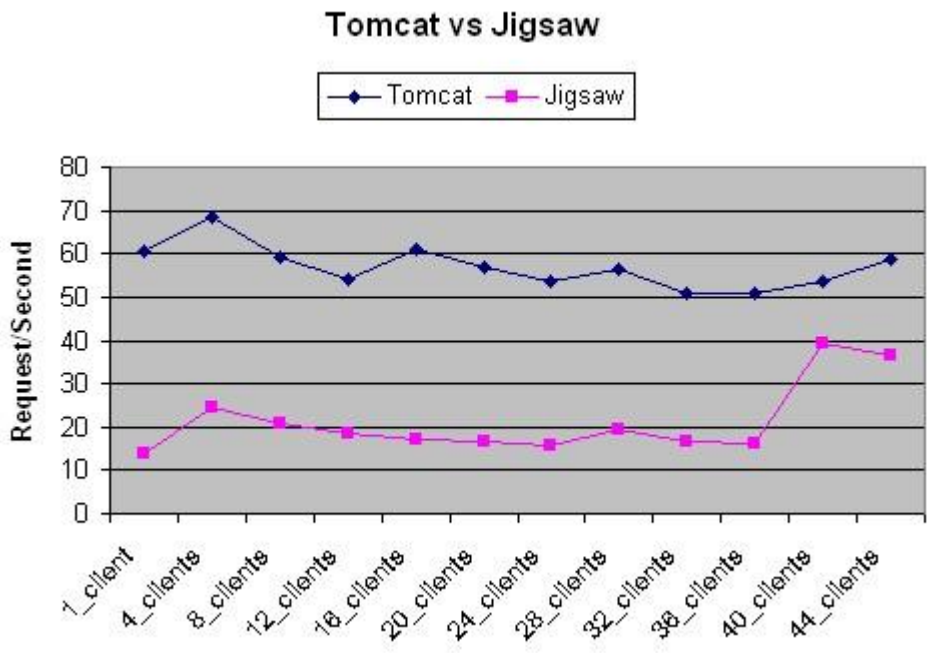


Figure 9. Tomcat/Jigsaw Benchmarking Results on One CPU

When we scale the test over eight CPUs, Apache 2.08a was more consistent in its performance, servicing more requests per second as we increased the number of concurrent clients without any fluctuations in the number of serviced requests (see Figure 10).

Num. of Clients	Apache 2.08a	Apache 1.3.14
1_client	176.417	187.017
4_clients	730.700	728.117
8_clients	1460.483	1446.617
12_clients	2197.383	2162.467
16_clients	2921.216	2869.700
20_clients	3642.583	3598.400
24_clients	4363.667	4300.117
28_clients	5071.800	5088.100
32_clients	5767.833	5821.950
36_clients	6442.083	5124.150
40_clients	6701.150	5119.900
44_clients	6721.317	5085.717
48_clients	6743.100	7092.250
52_clients	6775.300	7092.833
56_clients	6808.617	6570.800

Figure 10. Apache 2.08a/Apache 1.3.14 Benchmarking Data on Eight CPUs

Figure 11 clearly shows how consistent Apache 2.08a is compared to Apache 1.3.14. On eight CPUs, Apache 2.08a was able to maintain an average of 4,434 requests per second vs. 4,152 for Apache 1.3.14, a 6.8% performance improvement.

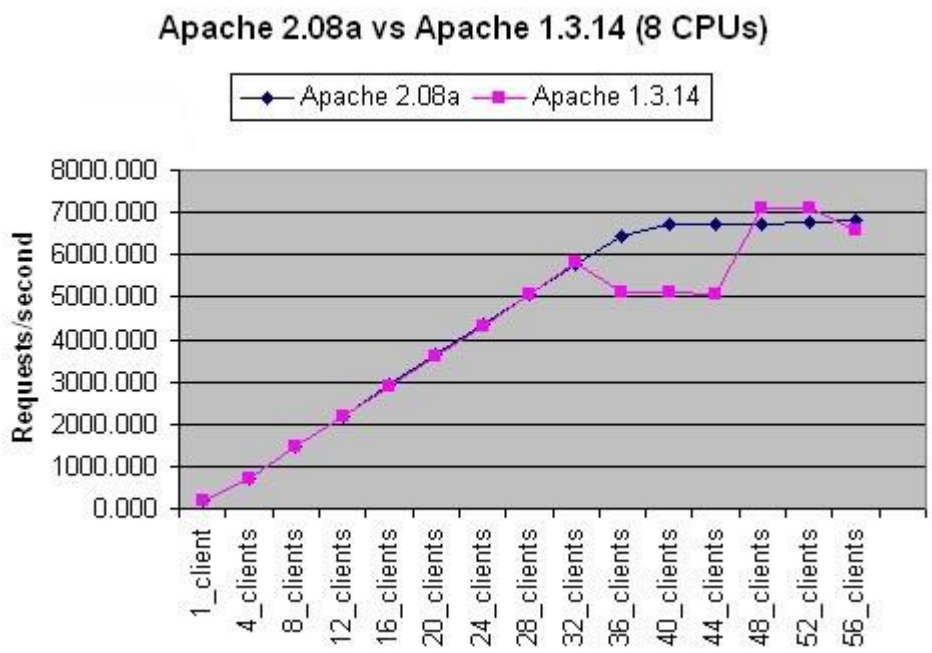


Figure 11. Apache 2.08a/Apache 1.3.14 Benchmarking Results on Eight CPUs

Scalability Results

We collected graphs for systems with 1, 2, 4, 6, 8, 10 and 12 Linux processors. For each graph, we recorded the maximum number of requests per second that each configuration can service. When we divide this number by the number of Linux processors, we get the maximum number of requests that each processor can process per second in each configuration.

Figures 12 and 13 show the transaction capability per processor plotted against the cluster size for both versions of Apache. In both figures the line is not flat, which means that the scalability is not linear, i.e., not optimum.

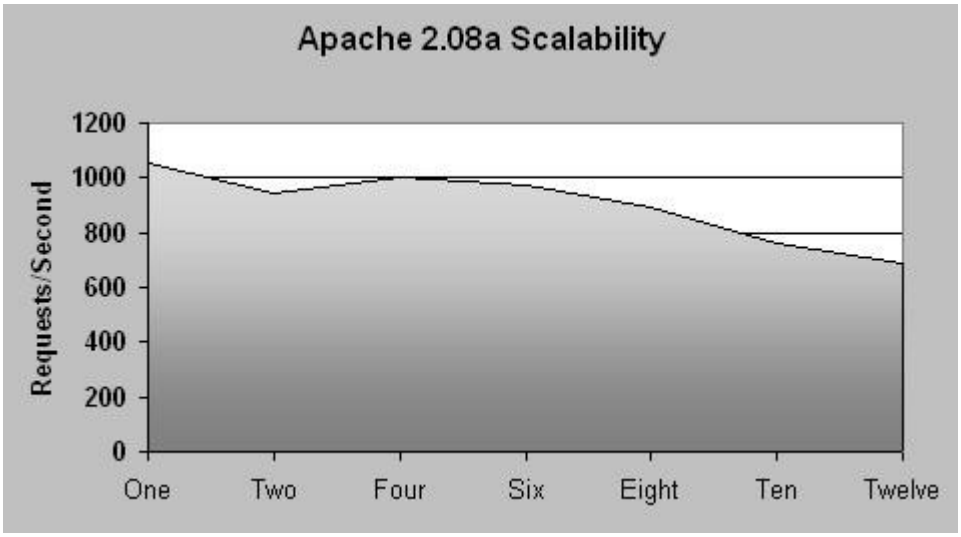


Figure 12. Apache 2.08a Scalability Chart

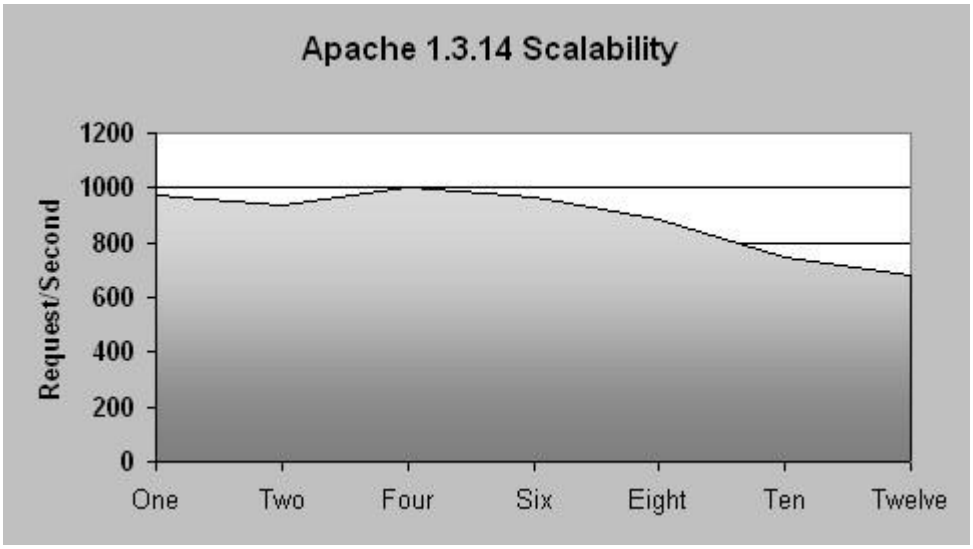


Figure 13. Apache 1.3.14 Scalability Chart

If we collect the scalability data of Apache 1.3.14 and 2.08a (see Figure 14) and create the corresponding graph, Figure 15, we observe that both servers have similar scalability compared to each other.

	1.3.14	*2.08a*
One	971	1053
Two	937	945
Four	1000	1003
Six	964	974
Eight	886	892
Ten	750	764
Twelve	683	685
Avg.	**884**	**902**

Figure 14. Scalability Data Comparison

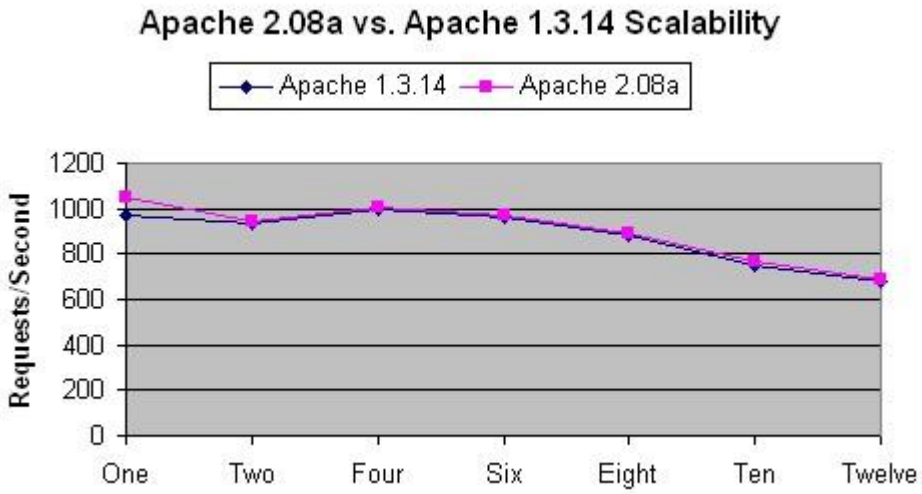


Figure 15. Apache 1.3.14 vs. 2.08a Scalability

On Linux systems both versions of the server have similar scalability. According to our results, Apache 2.08a is around 2% more scalable than the 1.3.14 version. In either case, we have a slow linear decrease. The more CPUs we add after we reach eight CPUs, the less performance we get per CPU.

As for the Java-based web server, although Tomcat showed a better performance (servicing more requests per second) than Jigsaw, it showed a slight scalability problem. Figure 16 shows a slight decrease in performance per processors as we add more processors.

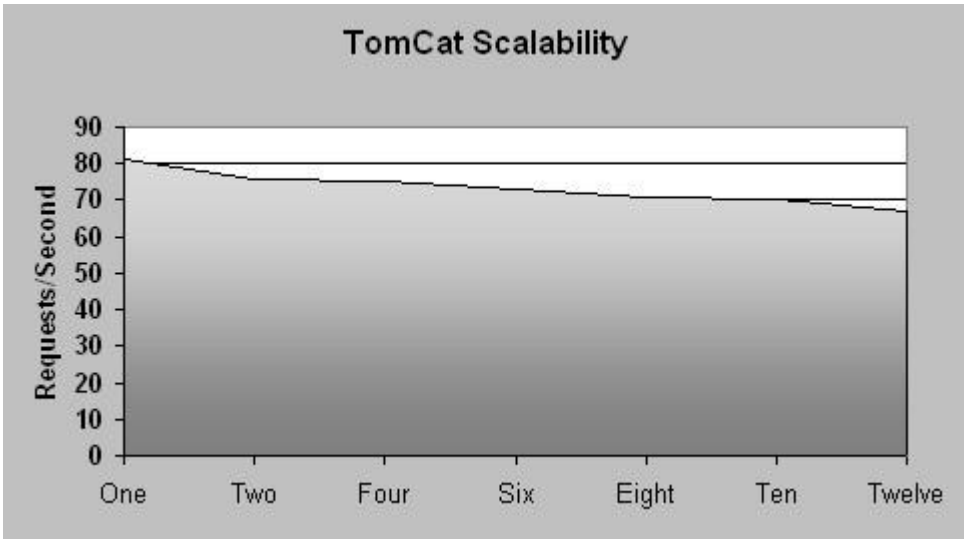


Figure 16. Tomcat Scalability Chart

Nonetheless, there are many possible explanations for the scalability degradation with the addition of more processors.

Factors Affecting Results

Several factors could have affected the results of the benchmarking tests:

1. We used NFS to store the workload tree of WebBench to make it available for all the CPUs. This could present a bottleneck at the NFS level when hundreds of clients per second are trying to access NFS-stored files.
2. Jigsaw and Tomcat are Java-based web servers, and thus their performance depends much on the performance of the Java Virtual Machine, which is also started from an NFS partition (since the CPUs are diskless and share I/O space through NFS).
3. To generate Web traffic, we were limited to only 16 Celeron rackmount units. The generated traffic may not have been enough to saturate the CPUs, especially in the case of Apache when we were testing more than six CPUs.

Problems Faced

During our work on this activity, we faced many problems ranging from hardware problems and working on prototyped hardware to software problems, such as supported drivers and devices. In this section, we will focus only on the problems we faced while completing our benchmarks.

We suffered stability problems with the ZNYX Ethernet Linux drivers. The drivers were still under development; they were not production-level yet. After reaching a high number of transactions per second, the driver would simply crash. The following is a sample benchmark on one CPU running Apache 2.08a. Once the CPU reaches the level of servicing 1,053 requests per second (throughput of 6,044,916 bytes per second), the Ethernet driver would crash and we would lose connectivity to the ZNYX ports (see Figure 17).

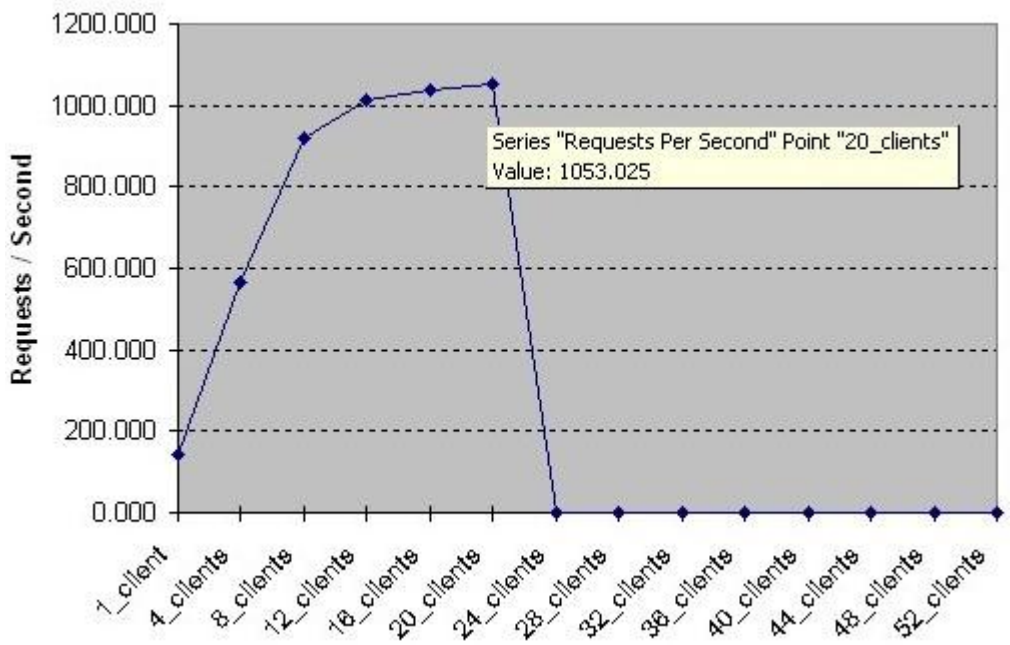


Figure 17. Ethernet Driver Crashing on High Load

We did much testing and debugging with the people from ZNYX and we were able to fix the driver problem and maintain a high level of throughput without any crash.

The second problem we faced when booting the cluster is related to inetd. The inetd dæmon acts as the operator for other system dæmons. It sits in the background and listens to network ports for incoming connections. When a connection is made, inetd spawns off a copy of the appropriate dæmon for that port. The problem we faced was that inetd was blocking for unknown reasons on UDP requests, and we needed to restart the dæmon every time it blocked. We are still having this problem even with the latest release of xinetd.

Another issue we faced was that we were not able to saturate the CPUs with enough traffic. That was obvious. We needed more power than what we were trying to benchmark. At the time we conducted this activity, we only had 17 machines deployed (one controller and 16 clients) for benchmarking purposes. It could be one reason why we were not able to scale up. However, we have increased the capacity of our benchmarking environment to 63 machines, and now we will be able to rerun some of the tests and verify our results.

Conclusion

ARIES started as a proof-of-concept project to study if we could build an internet server that has near telecom-grade characteristics using Linux and open-source software as the base technology. We have experimented with the various Linux distributions, web and streaming servers, traffic distribution and load-balancing schemes, distributed and journaling filesystems suitable for HA Linux clusters and redundancy solutions (NFS, Ethernet, Software RAID).

For the future, the work in ARIES will be directed toward augmenting the clustering capabilities of Linux to enable the system to accommodate more types of mobile internet services in addition to the already deployed web server applications looked at thus far. The focus will be to enable the system to reach the optimal utilization of the cluster's resources and to enhance the security aspects required within a mobile internet server. In addition, the project will augment the capabilities of the existing systems by supporting IPv6 technology.

We are keeping all three web servers on our experimental Linux cluster platform. The tested web servers did not scale linearly as we added more CPUs. However, they demonstrated very good performance and near-linear scalability (testing was limited to 12 CPUs). We are currently deploying the latest versions of Apache (2.0.15a), Jigsaw (2.2.0) and Tomcat (3.2).

Based on our tests, we believe that Apache has shown to be considerably faster and more stable than other web servers. We are looking forward to testing and experimenting with the 2.0 release version, which promises a clean code, a well-structured I/O layering and a much-enhanced scalability.

Acknowledgements

The author would like to acknowledge the Open Architecture Research Department at Ericsson Research for approving the publication of this article, as well as Marc Chatel and Evangeline Paquin, Ericsson Research Canada, for their help and contributions to the benchmarking activities.

Resources

email: ibrahim.haddad@lmc.ericsson.se

Ibrahim F. Haddad (ibrahim.haddad@ericsson.com) works for Ericsson Research, the Open Architecture Lab in Montréal, researching carrier-class server nodes in real-time all IP networks. He is currently a DSc candidate in Computer Science at Concordia University.

Archive Index Issue Table of Contents

Advanced search

Moving to PostgreSQL's Object-Relational DBMS

Chris Volpe

Issue #91, November 2001

Real-world advice on how to move an existing MicrosoftAccess database, one end at a time.

New World Tourist

The Hanz Scholz' Signature Twin Air Tandem with a Blue Fade

As more companies dive into open-source business systems, many are building web-to-back-end stacks that typically include Linux, PHP, Apache web server and an open-source database, usually either MySQL or PostgreSQL. PostgreSQL is gaining favor of late in many quarters, having reached, some say, a critical mass of functionality and stability. Several high-profile companies now provide 24/7 support, including Red Hat. This article shows what to expect when switching from Microsoft Access to an open-source database (in this case, BSD-style licensed PostgreSQL).

Michael Calabrese, manager of information systems for Bike Friday, recently undertook the challenge. Bike Friday is a rapidly expanding touring and mountain bike company based in Eugene, Oregon. It uses PostgreSQL to handle all of its sales, manufacturing and customer support data. Calabrese is also in the midst of changing all of the company's e-commerce systems from proprietary to open-source—Linux and Apache, with PostgreSQL at the core. For now, however, he has retained Microsoft Access 97 as the front end in order to minimize downtime while replacing the back end with PostgreSQL and adding new features. Calabrese says:

> If you're not dealing with preserving an existing front end, life is easy. Just run the conversion scripts [detailed below] and start writing a new front end. If you have an Access front end that you can continue to use with a PostgreSQL back end, you've provided clear pathways for things to grow, without trying to convert the whole system at once. In the first scenario, you'd be looking at a year for the conversion after freezing the whole front end. Attacking the change incrementally allows you to start designing new things with the choice of whether to do it in Access or PostgreSQL.

Calabrese decided to move to PostgreSQL because it is the more enterprise-capable system. It has a mature transaction-management system with a sophisticated data-locking mechanism called multiversion concurrency control (MVCC), which allows read-only access to data even if it's in use.

Tools of the Trade

Loading the Microsoft open database connectivity (ODBC) drivers onto the PostgreSQL template database lets Access and PostgreSQL set up house together. Besides basic conversion tools (see Resources), additional ODBC server-side functions that Access sometimes needs to run psql <*database name*> <*odbc.sql*> are located in the directory src/interfaces/odbc/odbc.sql. PostgreSQL also provides a platform-independent Type 4 Java database connectivity interface (JDBC) driver. An embedded interface for C (ECPG) is also part of PostgreSQL. Once the installation was finished, Calabrese chose data migration tools like pgAccess, available in Windows and UNIX versions, and exSQL public version 3.1.

After backing up existing databases using the included tools (either the pg_dumpall utility or a combination of the pg_dump and pg_dumpaccounts utilities) and running the Installer, the first step in the data conversion is to hunt down illegal file names in Access. Access is quite liberal in its allowance of illegal characters that other databases—Oracle, Sybase and PostgreSQL included—will not understand. Therefore, scores of illegal terms for Bike Friday's shipping

and ordering data that Access thought were fine had to be converted for PostgreSQL. For example, tables like Order Detail needed to become Order_Detail or OrderDetail, and field names such as Shipped? had to become Shipped or ShippedYN.

The basic conversion tools will remove all illegal characters automatically. This can be problematic for those working with an existing front end, because the front and back ends can cease communicating without an apparent reason. Calabrese recommends that anyone planning to preserve an existing front end should not change the names containing illegal characters in the front-end data or, alternatively, make parallel changes manually. In his situation, Calabrese found himself manually changing characters one by one on Bike Friday's front and back ends, which was okay since he was going to have to change the front end anyway. Either way, it's at this point that one should perform the first of many tests to be sure everything works. With illegal character issues resolved, the data is ready for conversion.

Converting the Data

For those planning to run an Access front end atop a converted back end, pgAdmin should do an adequate job of moving the data automatically. Calabrese also used a modified version of exSQL to define how Access and PostgreSQL would handle relationships between tables. The version he has made public at www.geocities.com/musica_6898/ postgresaccess_home.html runs a script that alters field-type conversion for several tasks, such as regulating how Access handles the money type. Bike Friday's Access front end saw PostgreSQL's numeric decimal fields as text fields. In order for Access to view the math properly, Calabrese changed the fields to a Float4—the method by which PostgreSQL describes a four-byte floating number—allowing Access to read them properly.

Testing the Front End

With more than 100 tables, Bike Friday's interface is fairly complex. Viewed from the user end, Bike Friday uses more than 80 screens for everything from entering an order, viewing a parts table, to scheduling production and tracking inventory. Therefore, Calabrese had to be sure that the system scaled for tens of users. Testing took several weeks, redesigning SQL queries as needed along the way, either by rewriting them on the Access side or, when that proved problematic, rewriting them on the back end until they ran at speed. Listings 1 and 2 illustrate the difference in typical queries and queries optimized for speed.

Listing 1. A Fairly Inefficient Query

Listing 2. The Same Query Optimized for Speed

Generally one optimizes PostgreSQL queries using SQL commands such as Create index, vacuum, vacuum analyze, cluster and explain. However, Calabrese offers this warning: Access 97 took the liberty of changing his queries based on how it thought they would be more efficient. Calabrese headed this off by using a pass-through query that told Access not to touch the query but send it straight to back end.

In the optimization he did for Bike Friday's PostgreSQL database, Calabrese scored most of his speed gains by extracting smaller, more exact amounts of data. Instead of the database querying 100,000 product order details at once, he told it to only look at the orders using some 2,000 details instead. “Access is greedy”, Calabrese said. “It grabs all the records and goes through them every time. That's very inefficient. We have 30 people with the company now, and if each has a computer accessing the database, that's going to be problem real fast in terms of speed.”

Troubleshooting PostgreSQL

The next stage in the changeover is debugging queries, and there are two basic routes here. The first is to activate and use the debugging tools in the ODBC driver for PostgreSQL. One can have the driver create a log so that whenever Access sends an SQL command, PostgreSQL puts it into the log, which is written to the root of the C drive. This will catch Access in the act if it tries to retrieve something like 100,000 rows or otherwise butcher a query and, for example, break it into a thousand smaller ones. Basically, it's an audit trail that makes it easier to catch haywire queries and rewrite them if something goes wrong, as it did here:

```
conn=86311032, query=' '
CONN ERROR: func=SQLDriverConnect,
desc='Error from CC_Connect',
errnum=105, errmsg='The database does not exist
on the server or user authentication failed.'
```

Alternately, if Access is sending a query and the system hangs, one can change the debug level on the server side to read the queries being sent to it. Fine tuning is a matter of going through each screen and testing them to ensure they're all up to speed by simplifying queries, making them faster or combining them. This process sounds easy but isn't, considering how esoteric some SQL lore can be. But, running two or three complete alpha tests at this point is going to save grief later.

The next step before putting the whole thing into production is to beta test it. Calabrese monitored Bike Friday's back-end system while salespeople, executives and associates used it in real time. “You're not just testing whether

the front end has errors, but how big you need to make the server", Calabrese said. He wrote a query script that kept close watch on the three main bottlenecks (CPU, the disk and the network) to see what loads they were taking.

For hardware tweaking (CPU, disks and memory), Bruce Momjian's *Linux Journal* article "PostgreSQL Performance Tuning" (August 2001 issue and also accessible on-line at www.linuxjournal.com/lj-issues/issue88/4791.html) provides a handy overview. Calabrese's script measured CPU stress based on how many seconds the load remained at 100%, 50% and idle. It looked at disk transfers in terms of the number of reads and writes to and from the disk, as well as the amount in kilobytes of those reads and writes. As for the network rate, Calabrese's script counted packets per second and bytes per second. Calabrese also suggested doing a ping/F on an isolated network, a flood ping that will indicate how much the server can take before it maxes. As far as memory goes, the more you have, the more data PostgreSQL will load into it and the faster the database will be.

Of course, the only way really to determine if a database is fast enough is whether or not the people using it feel it is fast enough. Fractional waits that seem insignificant on paper can be much longer in real time. Each organization will have its own tolerance level for speed and performance. The only way to be sure the database is working the way the organization wants it to is to let people use it and listen to what they say.

Finally, once you've gone through a few production tests, having listed and cleaned up all the errors in the interface, you're ready to roll out an open-source foundation for a real-world, enterprise e-business.

Resources

email: chris@macnet2.com

Chris Volpe is a technology writer based in New Hampshire. He can be reached at chris@macnet2.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Scalable Test Platform

Nathan Dabney

Issue #91, November 2001

Testing may not be fun, but it's important; Open SourceDevelopment Lab and Scalable Test Platform want to help.

The Open Source Development Lab (OSDL) is a nonprofit company working to enhance Linux scalability and telco capabilities. OSDL sponsors (www.osdlab.org/sponsors) have financed a full-scale test and development lab, complete with terabytes of storage and an array of SMP servers with anywhere from 2 to 16 CPUs. At the lab we provide developers with full access to enterprise-class machines via remote login.

We have been working with developers on the creation and execution of their tests. During this process, we have noticed a number of things that have to be done again and again for each test that comes through the lab. We listed the tasks that went into running an average test sequence and found a great deal of the process involved human interaction that could be automated. The Scalable Test Platform (STP) is the result of our attempt to automate the testing process from request to report.

Problems with Testing Methods

Benchmarking itself has inherent concept problems that are outside both the scope of this article and the scope of the Scalable Test Platform effort. There are, however, solvable problems with current testing practices, and that is what the STP attempts to address. Please keep in mind, the benchmarking we focus on is completely different from methods used to get marketable benchmark numbers.

The configuration of a testing environment is rarely as well documented as it should be. Documentation on the setup of systems used in tests is usually limited to what the tester believes is relevant to their specific research goals. This lack of detail will cause problems later on, when other analysts are

examining the report. It is not uncommon for an analyst to have to duplicate an entire test sequence to get the data required to answer questions that come up later. It is also common practice for a testing setup to be only partially automated. The resulting human interaction at undocumented moments will also affect the repeatability of the results.

Performance testing can require massive resources, both in the form of time and hardware. How many open-source developers can get access to 50 two-way client servers on a gigabit network in order to test a server farm made up of multiple 8-CPU servers and a 16-CPU server? Few companies would stretch to provide access to hardware like that and then only with a full entourage of managers and the potential revenue return to justify the expense. A good idea conceived by a developer without access to hardware like this is likely to remain unexplored.

Currently no central archive exists of well-documented results for performance, stability and standard compliance tests. Researchers are forced to run their own tests or pick and choose from mediocre results to come up with a less-than-accurate guess. System administrators have no central place to look for starter information on what combination of kernel, distribution and hardware tends to work well for a workload similar to what they anticipate. This lack of available research leads to confusion regarding the performance and reliability among the myriad of Linux choices.

Linux Kernel Development

Linux kernel developers cannot spend the time and effort required to run long performance and stability tests on their patches. Even if a developer is willing to spend the time testing a patch, testing software often requires a great deal of knowledge and specialized hardware just to install and configure. Occasionally this situation leads to problems being introduced into both the stable and development kernel trees. It also can allow problems solved previously to recur in future development but go unnoticed because of a lack of regression testing.

A number of developers have spoken up on the Linux kernel mailing list requesting a standard testing procedure for new patches. Many users and developers agree that a simple procedure, including performance, stability, standards compliance and regression testing, would benefit Linux kernel development.

While you can't test for every bug out there, you can check for common types of problems. It's generally not too difficult to add a regression test case to your testing suite after a bug is found and fixed. The problem is not in the creation of these tests. Most developers realize that it's a good idea to have a few synthetic tests available and very often do so. The problem is that most

developers can't or won't take the time to configure a full range of verification tests. While coding can be fun, testing is often quite boring. If a developer could easily request a full test of their code and then continue working while someone else does the dirty work, we think they would be more inclined to attempt verification runs on their patches.

The Scalable Test Platform

The STP is a hardware and software configuration for automated testing. To run the back-end control and scripting of the tests, we developed Brimstone, a combined batch control system and automated test harness. Requests are submitted using Eidetic, a user-friendly web front end, or brim-gate, the e-mail gateway. Using these front ends, entire test sequences can be requested in less than two minutes.

How STP works begins with developers checking patches in to our kernel CVS tree and requesting a test sequence using their patch. After the tests are completed, detailed results are returned to the developer via e-mail and are also archived on our web site. To simplify the process even further, a developer could write a short shell script that, in less than five lines, would check their patch into CVS and submit a preformatted test request via e-mail. Then all it would take to check the effectiveness of their patch would be a single command. Everything involved in a full-scale test run would then be taken care of without a second thought.

The hardware dedicated to the STP by the OSDL includes a 1.8TB storage away network setup connected to each server (four-CPU and up), via multiple Fibre Channel connections. Servers include two each of 2-CPU, 4-CPU and 8-CPU boxes, as well as a single 16-CPU IBM NUMA-Q server. A second 16-CPU NEC AzusA server, containing Itanium CPUs, is on order. We also have over 50 client-load machines that can be moved into the STP at the press of a key. We are also looking into the possibility of including a few single-CPU machines to ensure kernel modifications don't adversely affect the vast majority of current installs.

Eidetic, Brimstone and the e-mail gateway are all under the GPL, so interested parties can use them when setting up their own labs for specialized testing interests.

Requesting a Test Run

The first step is to go through the free sign-up to become an OSDL Lab Associate, available at www.osdlab.org. Next, enter a test request through the web page, which will involve something like this: choose the kernel tag to run (2.4.8 for instance), choose the distribution to use, choose the test to run, list

the CPU details, list the various hardware restrictions (optional) and enter an optional LILO command line (allows for restriction on the RAM used). After submitting the base items, you need to spend a moment filling out the setup page for the test you selected, then submit the final request.

It's as simple as that. Depending on the length of time required to complete the type of test requested, you could have your response back in less than 25 minutes. Full environment documentation and the resulting data sets will be archived on the web site for you. Short tests will take at least 15 minutes because a fresh copy of the OS is installed prior to every test sequence.

Since the entire process is automated, testing does not stop when our office closes. This means the quick response will be possible at any time, for developers located anywhere in the world.

Test Details

Since the STP is currently in the initial rollout phase, the number of scripted tests is still low. At the time of this writing, the following is a sample list of possible workloads: Juan Quitela's "memtest" VM abuse test suite, dbench (Samba) filesystem punishment, the ever-popular scripted kernel compile, simulated real-world CVS punishment and lmbench.

A long list of potential tests including scenarios involving multiple servers and applications such as Apache and MySQL is currently being evaluated. Of course, as an open-source project, we welcome assistance in the automation of these tests. Getting a full range of tests ready for use with the STP is going to be a major undertaking, but we believe the benefits to Linux will be worth it. For me personally, this is where I hope to give something back to the Linux community that will make a positive impact.

We are also in active cooperation with developers from SGI and IBM in the Linux Test Project (LTP). One current goal is to enlarge the LTP's coverage to include both targeted and general workload simulation tests. The LTP kernel features test is almost ready to be automated, which will provide us with a large regression test suite, as well as a solid base for stability research. The LTP's future plans include the development of a number of self-contained tests that will make great testing targets for the STP.

email: smurf@osdlab.org

Nathan Dabney (smurf@osdlab.org) has been working with Linux since Slackware in 1994. He enjoys breaking bad concepts and going for walks in the rain with his fiancee.

Archive Index Issue Table of Contents

Advanced search

More Than Word(s)

Jan Schaumann

Issue #91, November 2001

Ways to deal with MS Word documents using Linux and tocreate platform-independent documents.

More Than Word(s)

Ways to deal with MS Word documents using Linux and to create platform-independent documents.

by Jan Schaumann

We are told that Microsoft Word files can be viewed in any text editor, which probably is why so many people insist on sending even simple text documents as big Word attachments: “These download files are in Microsoft Word 6.0 format. After unzipping, these files can be viewed in any text editor, including all versions of Microsoft Word, WordPad and Microsoft Word Viewer” (from Microsoft's web site). How often do you receive an e-mail with a Word document attached because the sender simply assumes that everyone uses Microsoft Word (presuming they think about it at all)? Not only is it dangerous for anyone actually using Word to open attachments that might possibly contain macroviruses, but for anyone not using any of Microsoft's products, they have become a real nuisance. Actually, even people who do have MS Word need to make sure they have the latest version (and possibly buy an upgrade) because the discrepancies between the various versions are so large that sometimes Word can't read Word. This article tries to make your (office) life a bit easier by elaborating on the various possibilities of dealing with these dreaded documents.

Short of rejecting any documents not in standard format (more on this later), there is no optimal way of dealing with MS Word documents in Linux. As mentioned above, sometimes even Word can't read Word. There are, however, a number of approaches to opening most documents and even preserving the format.

There are the full-fledged word processors (very similar to Microsoft Word), a few file converters and some rather unconventional means of extracting the information out of a .doc file. Depending on your needs, you may choose different solutions in different situations.

Big Words

If you need to do a lot of word processing and often exchange documents with coworkers, you most certainly will want to install a complete office suite. An office suite comes with, among other things, a word processor that lets you read (and sometimes write) various MS Word formats, even though they all have their own document formats as well. The most common office suites available for Linux are Applixware Office, Corel WordPerfect Office 2000, KOffice and StarOffice (OpenOffice).

Applixware Office

Of all the above, Applixware Office is the only one not available at no cost. However, Applix was kind enough to provide me with a copy of their software for this article (retail price is $99 US). I received a few colorful boxes containing a copy of Applixware Office, Applixware Words (standalone) and Applixware Spreadsheets (standalone). The Office package came with a beautiful handbook, something that I would certainly appreciate, once I set it up.

Eager to test the new software, I attempted to install Applixware Words, following the instructions in the manual. As I run Debian, I do not mind the fact that the instructions to install per RPM are missing a step (you need to change the directories once more to find the RPM-install script, which was not mentioned), and I happily executed the setup binary.

At first, things seemed to go smoothly, but then closed-software practices took their toll. There appears to be a small bug in the install scripts that made it impossible for me to install the software. When I attempted to install into /opt/applix (as suggested by the program), the error log told me after the failed installation that it apparently tried to install to /optapplix, even though it created /opt/applix.

This would be just a minor nuisance if the user was able to edit the install script, but as this is closed software, there was nothing I could do. I tried /opt/applix and a few other tricks, but to no avail.

Due to unmet dependencies (on a Debian system, it appears as if I did not have any of the most basic RPMs installed) the RPM install fails as well, so that in my last attempt to install Applixware Words, I generated .deb packages from the RPMs using alien:

```
mkdir /tmp/applix
cp cdrom/RPMS/*.rpm /tmp/applix/
cd /tmp/applix/
alien *.rpm
...
dpkg -i --force-overwrite *.deb
```

This seemed to install the packages, but running the application led to several errors. I finally gave up and wrote an e-mail to Applix to inquire about these problems.

While word of mouth has it that this is a solid product—an advantage of this suite is that its native file format is plain ASCII text with the specifications freely available from the web site, which makes it easy to write import/export filters—these kinds of problems are rather frustrating.

The advantage of having a nice printed manual with a (supposedly) fine product is, in my opinion, outweighed by the fact that it's closed software. I can't go in and try to fix errors myself, which leaves me helpless.

Corel WordPerfect Office 2000

Corel, best known by most for its Linux distribution (Corel Linux) and CorelDRAW, also developed a very powerful office suite. Corel WordPerfect Office 2000 includes the well-known WordPerfect word processor. WP offers anything one might wish for in a tool like this; it has been regarded as superior to MS Word by many people and is available for Windows and Linux. Curiously, it is not available for the Mac, even though Corel offers other software for this platform.

If you want to license WP Office 2000 for your entire office, you will find a hefty price tag attached; however, for personal use, you can download WordPerfect by itself for free. I found it a breeze to install and use; it easily opened all of the Word documents I found on my hard drive and was even able to display mathematical formulae properly.

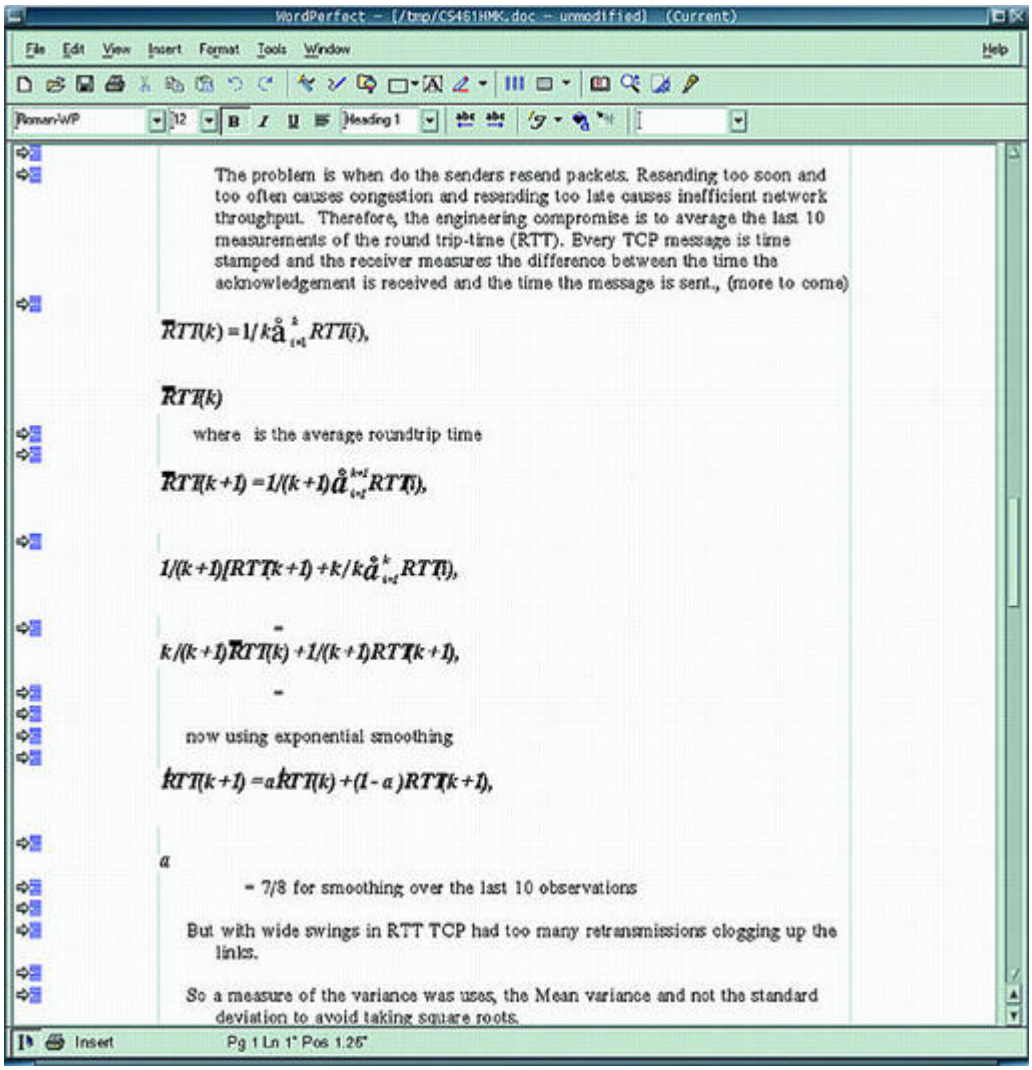


Figure 1. A Screenshot of WordPerfect 8

KOffice

KOffice, brought to you by the friendly people of KDE, was released together with KDE 2.0 in October 2000, as beta software. Nonetheless, the word processor KWord looks impressive. It integrates nicely with all the other KDE applications and neatly imported most of the MS Word documents I fed it.

Problems arose when I tried to open a document containing mathematical formulae, but since I have been assured that these formulae bring down every version of Word itself but the latest (no surprise there), I would still recommend it. By the time KOffice 1.1 will be released, I'm sure KWord will easily suffice for most needs.

This office suite is, of course, licensed under the GPL and available for free download from your favorite mirror. Debian's apt-get install, kword, took care of all dependencies for me, but since KOffice relies on KDE 2.0 and Qt 2.2, you might find yourself upgrading a lot of packages before you can use this program.

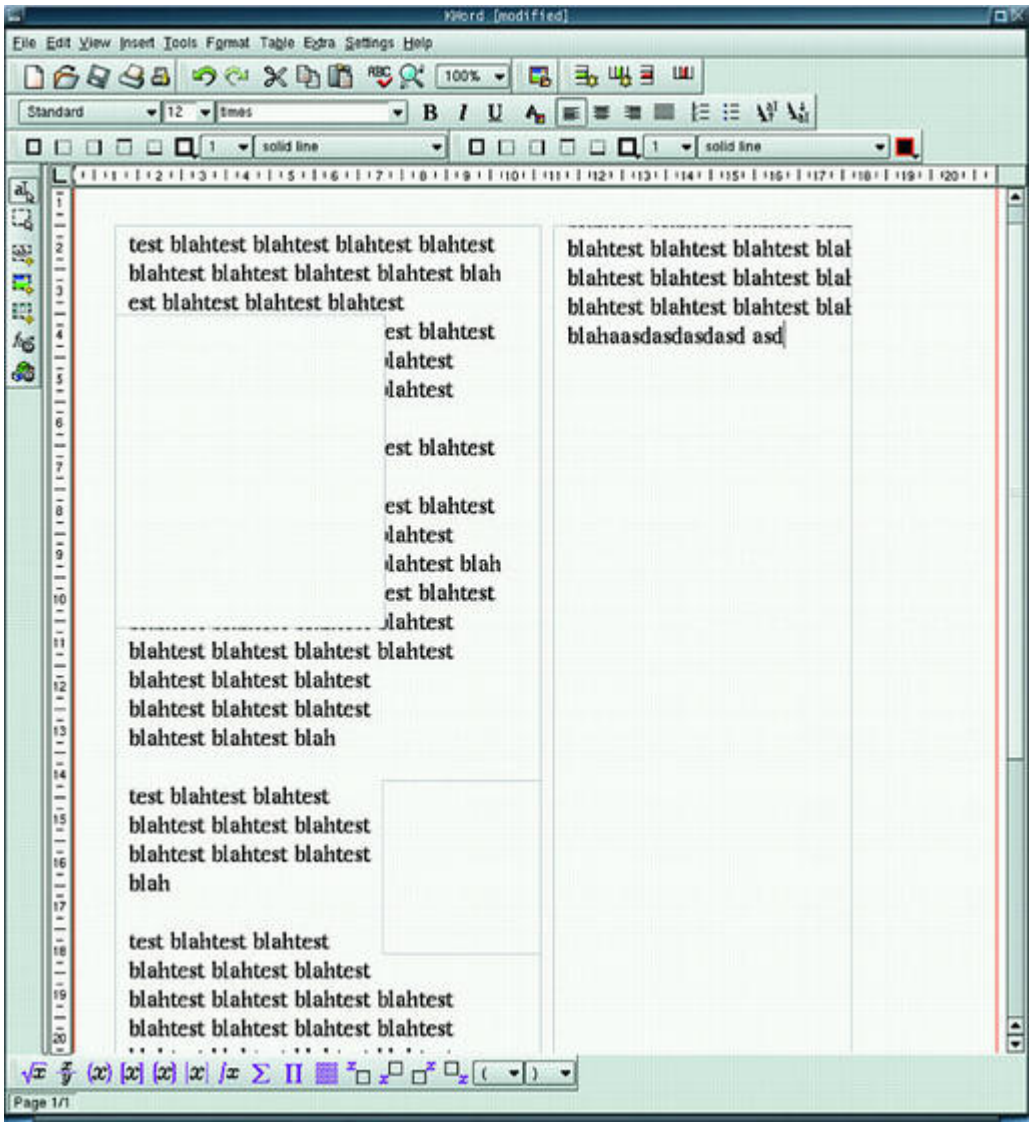


Figure 2. A Screenshot of KWord

StarOffice/OpenOffice

Some time ago, Sun Microsystems acquired StarOffice, an office suite available for many operating systems. StarOffice was one of the first office suites able to compete with Microsoft's Office. While Sun always offered StarOffice for free download, only fairly recently did they announce the release of the source code to the Open Source community, which ultimately led to the OpenOffice Project. So, this is another GPLed Project.

StarOffice/OpenOffice includes a very powerful word processor, which can read most Word Documents and can even write to .doc format. However, it has a drawback: it's a memory hog. Not only does it require a significant amount of disk space for the complete installation, it also takes awhile until all the components are started. If you have a slow machine, this might not be your first choice. On the other hand, if you have enough space and memory, I'm sure you'll find that StarOffice/OpenOffice is able to meet all of your word processing needs.

Small Words

All of the aforementioned applications are full office suites, rather hefty packages more suited to people who actually do perform a lot of word processing and who, at the same time, need to have applications for spreadsheets and presentations, etc.

For those of you who just want a word processor for the occasional letter of complaint to your landlord, there are some lighter approaches. The most common lightweight word processor is AbiWord. AbiWord, designed to be “full-featured and remain lean”, seems to live up to its goal. It's fast, available for a large variety of platforms, free (as in beer and as in speech) and under heavy development. However, I do have to admit that it chokes on some documents or opens them without preserving the original format. In particular, MS Word's way of dealing with tables seems to confuse AbiWord.

Another very small and light word processor is Pathetic Writer (pw), which is part of the Siag Office Suite. The reason I mention pw here and did not include it with the full-fledged office suites is that it seems rather thin. **pw** will not open Microsoft's .docs, but it will happily perform your everyday word processing and can import and export most common formats. Siag Office, just as AbiWord, is published under the GPL and is available for free download.

Unconventional Words

All of the above-mentioned applications have various requirements: some rely heavily on pre-installed libraries (such as KWord), some are rather resource hungry (StarOffice/OpenOffice), others are expensive and/or not open sourced. However, all of them try to preserve a certain style or a certain way of formatting a document.

While this is certainly useful and important, I have found that I have no use whatsoever for a word processor, no matter which one. In 90% of the cases where some thoughtless person sends me a .doc file, the information contained within the document could have been communicated easily in plain text in a fraction of the file size.

So let's talk business now and see how we can extract the necessary information from proprietary file formats. There are a few tools worth mentioning, and their beauty lies in the fact that we do not even need X because they are all command-line tools.

antiword

antiword takes a Word document as input and extracts the information contained in it, converting it to plain ASCII text or to PostScript. It tries to maintain the formatting as much as possible, and it does a fairly decent job of it.

It's quick, and because it is a command-line tool, we can redirect the output to another process or file for further modification. To take a quick glance at the content of the file, you could pipe the output to less:

```
antiword HUGE.DOC | less
```

Or, if you'd rather have a hard copy:

```
antiword -p letter HUGE.DOC | lpr
```

I found antiword so useful that I replaced my previous mailcap-entry for handling of MS Word files (which I used to call abiword) with the following line:

```
application/msword;antiword %s | vim -
```

This allows me to read through .doc attachments from my mail reader (mutt), and since I pipe the output right into my favorite editor, I even can make modifications and save it to another file. Note that by placing this entry into my ~/.mailcap, all applications respecting this file will use antiword and vim to display .docs. If you are using a graphical browser such as Netscape, you might want to use a different editor or use the -g switch for vim to spawn a GUI front end.

If you are a hardcore minimalist, you will find that the command strings, part of the GNU binutils package, is often sufficient to extract the plain text information from a .doc file. However, antiword has the significant advantage over strings in that it can also extract images in addition to text.

For details on the use of the various options and on how to extract images from a Word file, see antiword at www.winfield.demon.nl/index.html.

wv

The other application, formerly known as mswordview and now available as wv, has been around for quite some time. When I first installed Red Hat 5.2 a few years ago, the Netscape browser used mswordview as the standard application to handle .doc files, as it converted them quite reliably into nice HTML. Note that I'm not talking about wordview, a Microsoft product. The similarity in the name caused the author to rename his tool.

While it certainly is great for a browser to use an application that turns Word files into HTML, this is not always the ideal output format. Therefore, wv now includes a whole set of tools to convert Word documents into a large variety of formats, including, but not limited to, ASCII text, HTML, LaTeX, PostScript and PDF. **wv** is published under the GPL and is available for free download.

No Words

So far we've seen how we can read Word documents and even what options there are to write documents that, in Winworld, would most likely be done in Word. But I can't help concluding that the word processor itself, as an application, is not required or is useless in the vast majority of cases.

The typical user trying to write a simple progress report, for example, usually follows a certain scheme: writes something, uses mouse to highlight the text, uses mouse to point and click and select bold, presses Return a number of times, presses Space a number of times, decides he/she doesn't like it, presses Delete a number of times, uses mouse to point and click and select italics, repeats.

I am fully aware that this is not the proper way to utilize a powerful word processor, but let's face it, that's exactly how the majority of users (those for whom these “user-friendly” applications are designed) work. The efforts required to enter a table of contents, a bibliography, cross-references, etc., can only be imagined.

Eventually, the outcome is a document that takes hours to prepare and that looks only the way it should on this platform using a particular version of this word processor. To avoid such bad practices, let's investigate some alternative methods for preparing platform-independent documents.

A Classic: ASCII

As I have mentioned repeatedly, the information contained within the majority of documents is plain text. At times some fancy formatting may be nice, but it's optional. The main interest of the person writing the document should be to communicate the information.

Simple, plain ASCII text is usually sufficient to send information from one person to another—that's exactly why e-mail, for example, is still a text medium. HTML in e-mails does not add anything to the content. ASCII text can be read from anywhere with any editor (and not just with “any editor, including Microsoft Word...”). By structuring the text clearly, by using paragraphs and horizontal lines constructed out of hyphens, maybe even by using *bold*, /

italics/ and _underlined_ text as used on Usenet, one can write clear, easy-to-read and understand and, most importantly, portable documents.

LyX and LaTeX

While plain ASCII text should be the choice for most cases, it cannot be denied that occasionally one might need or want more formatting. Well, no need to dig out the old word processor again. Just use LyX, the graphical front end to LaTeX.

LaTeX is an astounding typesetting engine derived from TeX. It takes a .tex file as input and typesets it, generating a .dvi file. It is available for a large variety of platforms, and documents typeset with LaTeX look incredibly professional. Yet, you can use your favorite editor to create the input files because LaTeX is a command-line tool.

When using LaTeX, one can concentrate on the content of the document instead of the way it looks because the typesetting engine will take care of the layout. A .tex file contains a few tags (which may remind you of HTML) to determine the way the text will be displayed.

This is a completely different way of writing a document from word processors; no more pointing and clicking and highlighting and reconsidering and so on. But, it may be daunting to someone who is used to using a GUI.

Now this is where the good guys from LyX come into play. They developed a GUI for LaTeX, enabling the inexperienced user to take advantage of the power of TeX without having to learn it from scratch (yet).

Upon first glance, LyX may look similar to your average word processor, but if you follow the tutorial, you will quickly see the difference and how you can increase productivity by concentrating on your work and material, rather than on the visual representation.

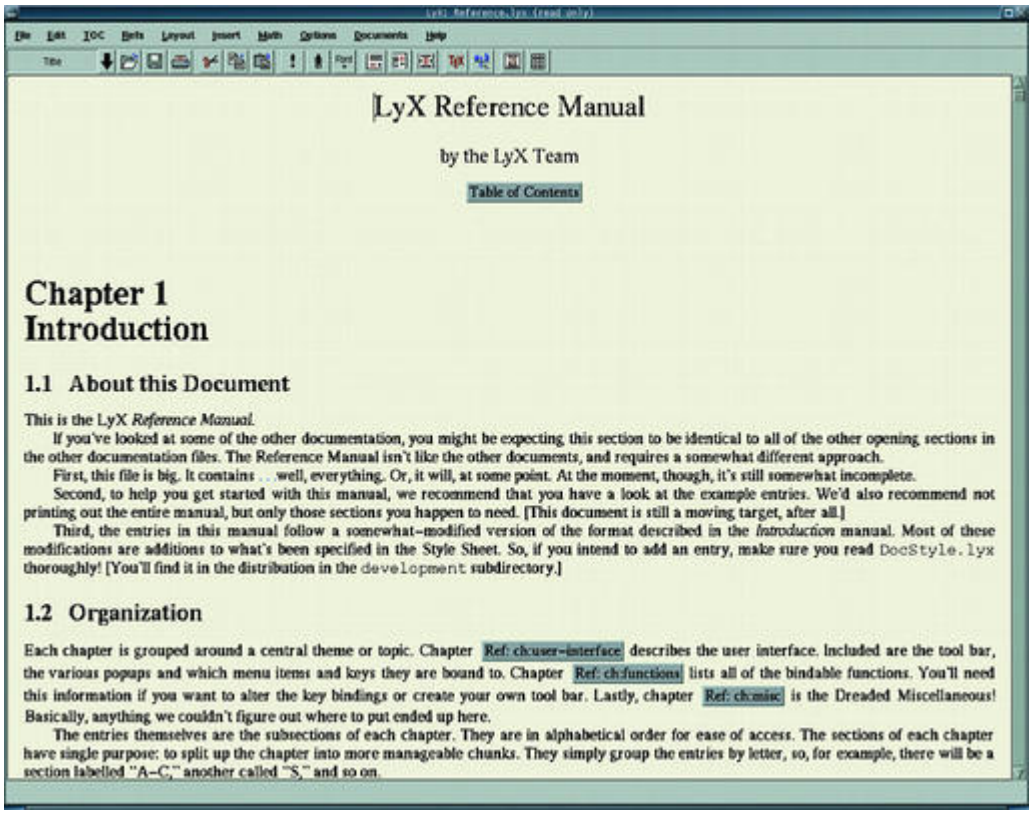


Figure 3. A Screenshot of LyX

If you often connect to your machine remotely to get work done, you don't always have the ability to export your display or to forward X. This is when you learn to appreciate the power of the command line—you find that everything you will ever need is right there at your fingertips. By using your favorite editor (vim, in my case) and LaTeX, you can get all your work done easily through a single terminal to your machine.

Another advantage of LyX and LaTeX is that you can easily export your files into platform-independent formats such as PostScript or PDF. By combining the power of **make** with the power of LaTeX, this can be done with just a few commands. Take, for example, this document—I turned the input file into a beautiful PDF (Figure 4) simply by using the command **make pdf**.

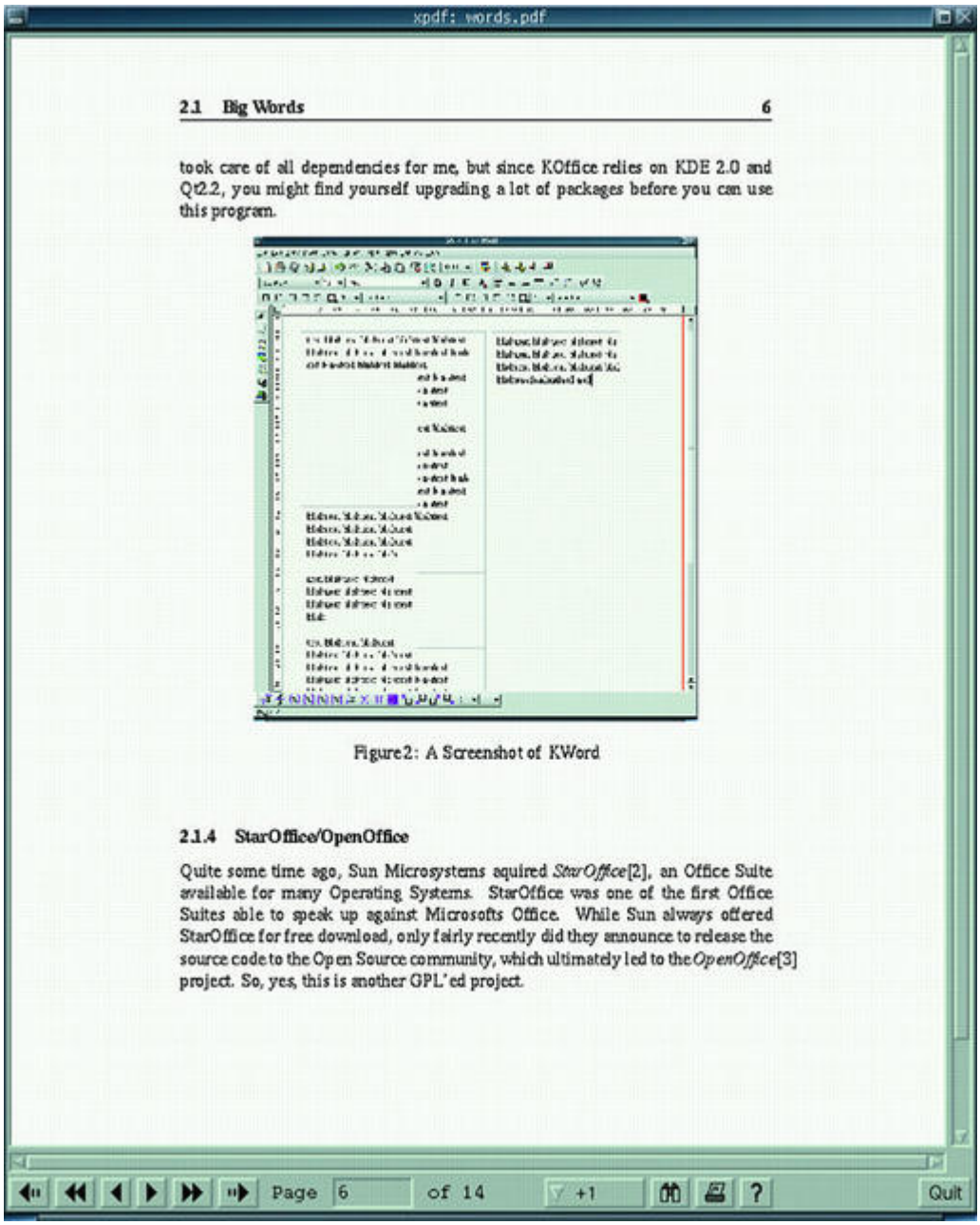


Figure 4. This Document in Source and in PDF Format

Even though the Makefile itself (see Listing 1) is simple, it allows me to convert my document easily into a large variety of output formats using several different command-line tools, such as ps2pdf and latex2html.

Listing 1. The Makefile for This Document

Finally, LaTeX is extensible—you can write your own styles to achieve different results depending on the kind of document you are writing. But most likely, someone else has already done so and uploaded it to the Comprehensive Tex Archive Network (CTAN, TeX's equivalent to Perl's CPAN).

Conclusion

In brief, whichever way you choose to handle your word processing, the importance of conveying the information in a portable document format needs to be expressed. Just try to make it clear to the people with whom you correspond, to the people who continually send you MS Word documents and then insist that you “fix your computer” when you tell them that you can't open them or that some formatting got lost. I have found that if one explains in a friendly way how a PDF or a PS, for example, can be read by anyone on almost every platform, occasionally one can educate all but the most stubborn citizens of Winworld.

Personally, I'm sure you will find that LaTeX is far superior even for these little everyday tasks when it comes to creating professional (looking) documents. In order to take advantage of LaTeX, however, it is necessary to free your mind from what you may be used to. This may take awhile, but don't be afraid, there is a lot of helpful documentation out there. Maybe the most important document for a LaTeX beginner might be “The Not So Short Introduction to LaTeX2”, available from Comprehensive TeX Archive Network (www.ctan.org).

After a short time of going through a tutorial and, most importantly, giving it a try and taking a look at some examples, you will never want to go back. You can take my Word for it.

Resources

email: jschauman@netmeister.org

Jan Schaumann (jschauma@netmeister.org) was born in Iserlohn, Germany. He grew up in Altena, Germany and studied for two years toward a Master's in Modern German Literature and Media and American Studies in Marburg, Germany. He moved to New York City in 1998.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Getting Your Palm to Talk to a Linux Box

Johan Coppieters

Kevin Velghe

Issue #91, November 2001

Two students develop a solution to help get you in sync withyour palm device.

A Palm is such a neat portable device. You take notes on it, plan meetings or even write down anniversaries. It's a nice tool to have while on the road. Our intranet server, on the other hand, is also a great resource. We have our company's planning and agenda on it, and it has all those interesting notes, addresses and, in our case, a complete knowledge database. This server runs (of course) Linux, an Apache web server and MySQL database on top of it, glued together with a proprietary application engine.

Wouldn't it be great if the two environments could be synchronized? Of course! Now if we could only access the internal databases from the Palm on our Linux server, that would get us pointed toward the right direction. Documentation is sparse and the Internet is big. But then along came Kevin and Jeffrey, two students who, with some effort, put together a solution that allowed us to make changes on the Palm or on the corporate intranet server and have these modifications propagated to one another's counterpart databases.

All code below was tested on an Intel machine running a Red Hat 6.x release and a Palm Vx with serial cradle running Palm OS 3.5, but other combinations shouldn't cause any problems. The libraries used are unchanged since the first Palm appeared on the market, when they were still being produced by 3Com. I suppose (but haven't tested it) that Visor's won't cause any problem either, as they are running the same OS.

Let's start with the simple part: connecting the Palm to the server. First, we need to connect the cradle of the Palm with the serial port of the server. Then we create a device called pilot. This pilot is nothing more than a link to a serial port (in our case /dev/ttyS0):

```
ln /dev/ttyS0 /dev/pilot
```

Now, we're ready to open the connection to the Palm using a C program and a simple push on the HotSync button of the cradle. Once the connection is made, we can relax and read records from the Palm's databases.

Communicating and opening the connection to the Palm is easy with the pi library. This library simulates the BSD socket interface: create a socket, bind it to our device, listen for an incoming connection and accept it. The incoming connection is initiated by the Palm/cradle combination when a user hits the sync button on the cradle. Making a dæmon that waits for someone to come by and synchronize its Palm is illustrated in Listing 1. Loop this program forever.

Listing 1. Synchronizing the Palm

Once the connection is open, how do we interact with the Palm databases? Each database on the Palm has a name. We can open a database by name and get a specific record or cycle through the database. On a Macintosh or Windows platform these functions are performed by conduits. Palm itself provides a conduit on these platforms for every standard database included with the Palm OS bundle. The Palm database manager provides ways for us to cycle only through the modified records in a database. Modified since when? Well, since the last synchronization, the last time a server committed this database. So, we should do this in our programs when finished with the synchronization process. The lines in Listing 2 should be executed when the connection is open.

Listing 2. Cycling through Modified Records

Reading records from a Palm database doesn't qualify as synchronizing. We'll need to do more, such as write to the Palm, delete from it and read from our MySQL database. Since connecting to a MySQL database is outside the scope of this article, we won't discuss the further details of the synchronization problem (and the possible conflicts). However, Kevin Velghe has written an excellent document about it. You can read it at www.duo.be/palm/mysql_palm.html.

Records stored in a Palm database have unique numbers. Whenever you write a record to the device, it returns this number. You should store it on your desktop or central database, so that you can delete or update a specific record.

The dlp_WriteRecord accepts a Palm ID. If it is zero, the Palm OS will allocate a new one for you; if you pass an existing ID, the corresponding record will be updated. For most standard databases a pack function packs the structured record into a buffer. This process is shown in Listing 3.

Listing 3. Pack Function

Identifying the Palm

If, as in our case, you're serving multiple Palms on one server/cradle, you'll need to find out whose Palm is in the cradle. Once the connection is open, call the ReadUserInfo function:

```
  {
    int db, len, I, attr;
    recordid_t id;
    struct PilotUser U;
```

...open the connection...

```
    sd = pi_accept(sd, 0, 0);
    dlp_ReadUserInfo(sd,&U);
    printf("Palm of: %s", U.username);
    pi_close(sd);
  }
```

Deleted Records

The Palm database manager does not delete records automatically when the records are read. It marks them for deletion and even has the ability to mark them for archiving on the desktop or server counterpart. When reading a modified record one should check the attribute flag to see if this record needs to be deleted (or archived). It will be deleted permanently on the Palm once the database is cleaned up:

```
  {
    ...
    for (;;) {
      len = dlp_ReadNextModifiedRec(sd, db,
                          buffer, &id, &I,
                          0, &attr, 0);
      if (len < 0) break;
      if ((attr & dlpRecAttrDeleted) ||
          (attr & dlpRecAttrArchived))
        printf("Marked for deletion: %ld", id);
    }
  }
```

Some Logging Won't Hurt

A good practice after synchronizing a Palm is to leave some comment about it in the Palm's log. You can write whatever you want, the time and date are added anyway. So, add the following code to the end of your programs:

```
  {
    ...
    dlp_ResetSyncFlags(sd, db);
    dlp_CleanUpDatabase(sd, db);
    dlp_CloseDB(sd, db);
    dlp_AddSyncLogEntry(sd,
      "Read modifications from Pilot.\n");
    pi_close(sd);
  }
```

What You Need to Start Working

To use your Pilot with a Linux box, get the pilot-link package. The interfaces exist for many platforms, from Next, BSD, Solaris, OS/2 to Linux. They let you write programs in many languages from Python, Java, Perl, Tcl to C++ and C. The FTP site ryeham.ee.ryerson.ca/pub/PalmOS has the file you need: pilot-link.0.9.3.tar.gz.

It compiles on a Linux box without trouble. It is really more than just an interface library containing a bunch of simple tools that illustrate its use. These simple tools are very useful, enough to back up a Pilot (and restore it), move data to and from it, send e-mail, install programs and databases and so on. The library is callable from C, C++, Perl, Python, Tcl and maybe a few other languages; if you happen to have any programming ability, you can craft tools to do anything you want, using the provided ones and the sample code in this article as examples.

New programs, extra documentation, remarks or HOWTOs may be submitted to palm@duo.be. We'll put them on the server available for the public at www.duo.be/palm.

Let's install the package by executing

```
tar -xvzf pilot-link.0.9.3.tar.gz
```

This will create a directory (pilot-link.0.9.3) containing the sources. Change your working directory to the source (pilot-link.0.9.3) directory.

Run **./configure**. This will search through your system for information needed to compile the software. Configure will set things up to be installed in /usr/local by default. If you want to change it, run **./configure --prefix=DIR**, where DIR is replaced with the name of the directory to which the software will be installed.

Run **make**. This will compile the software. The software will not be installed until later, so that you have a chance to try it out first. If you are replacing an older version with a newer release, you may wish to check to make sure that no functionality you need has been broken. Generally, this is not a problem.

As the root user, run **make install**. This will copy the software into directories under /usr/local (or wherever you specified with the --prefix option). If you cannot log in as root, you can install the software to some directory where you have write access.

Don't forget to add any new directories of executables to your search path. Check out all the neat tools installed together with the libraries. For a description of some of them, take a look at the article mentioned in Resources.

Resources

Johan Coppieters (johan@duo.be) runs a company, Duo nv, based in Bruges, Belgium that develops web sites, intranet and extranet applications for some of the bigger companies in Belgium.

Kevin Velghe (palm@duo.be) has written the synchronization C-program Palm-Linux and did some of the research while doing a three-month school/business exchange training. He can be reached at Duo nv.

Archive Index Issue Table of Contents

Advanced search

Building the Ultimate Linux Box

Eric S. Raymond

Issue #91, November 2001

Focusing on maximum crunching power and PC hardware whendeveloping the Ultimate Linux Box (ULB).

Five years ago, I wrote an article for *Linux Journal* that developed a recipe for an elegant and economical Linux box. The article became one of the most popular in *LJ*'s history, so the editors have invited me back for a second round.

The UL Web Server

This time around *LJ* recruited Rick Moen, author of some well-known FAQs on modems and other hardware topics, to assist. Daryll Strauss, the man behind the famous Linux renderfarm used in the movie *Titanic*, contributed sage advice from his background in graphics and extreme data crunching. Also, instead of going for economy we're going to go for maximum crunching power. We're going to ask how to get the absolute highest performance out of hardware we can live with.

Hardware you can live with means a machine that is stable, easy to troubleshoot and inexpensive to maintain. It should be small and low-maintenance enough to live beside your desk, not some liquid-cooled monstrosity. It should be, in short, a PC—a gold-plated hot rod of a PC but a PC nevertheless. Other important aspects of livability are the levels of emitted acoustic noise and heat; we'll be minimizing both.

We'll stick with PC hardware. Alphas are fast and have that wonderful, sexy 64-bit architecture, but the line seems all too likely to be killed in favor of the Itanium before long. Considered in isolation, I like the PowerPC chip a lot better than any x86 architecture. But PC hardware has all the advantages of being the biggest market; it's the easiest to get serviced and least expensive to upgrade.

The Ultimate Linux Box that we showcase will, of course, fall behind the leading edge within months (or even by press time). But walking through the process of developing the ULB will teach you things about system design and troubleshooting that you can continue to apply long after the hardware in this article has become obsolete.

Good News, Bad News and How Can You Justify This Thing?

What to Optimize

For typical job loads under Linux, the processor type is nearly a red herring—it's far more important to specify a capable system bus and disk I/O subsystem. If you don't believe this, you may find it enlightening to keep top(1) running for a while as you use your machine. Notice how seldom the CPU idle percentage drops below 90%.

If you're building a ULB, go ahead and buy the fastest available processor. Once you've gotten past that gearhead desire for big numbers, pay careful attention to bus speeds and your disk subsystem because that's where you'll achieve serious performance wins. The gap between processor speed and I/O subsystem throughput has only widened in the last five years, so this is even better advice than it was in 1996.

Everybody's Doing Dual Athlons

How does all this translate into a recipe in 2001? Get a PCI-bus machine, not a hybrid PCI/ISA design; you sacrifice about 10% of peak performance with those. Get the fastest available front-side (processor-to-memory) bus (in August 2001, the maximum is 266MHz). Get a high-speed SCSI controller and the fastest SCSI disks you can get your hands on.

The case for SCSI is a little less obvious but still compelling. For starters, SCSI is still at least 10%-15% faster than IDE/ATAPI running flat out. Because it's perceived as a professional choice, SCSI peripherals are generally better engineered than IDE/ATAPI equivalents, and new high-performing drive technologies tend to become available in SCSI first. You'll pay a few dollars more, but the cost is well repaid in increased throughput and reliability. Rick Moen comments:

> They call me a SCSI bigot. So be it—but my hardware keeps being future-proof: I don't have to run bizarre emulation layers to address CDRs, I never run low on IRQs or resort to IRQ-sharing, all my hard drives have hardware-level hot-fix, all my hard disk/CD/tape/etc., devices support a stable standard rather than this month's cheap extension kludge, and I don't have to

> worry about adverse interactions at the hardware or driver levels from mixing ATA and SCSI.

Neither Daryll nor I will have IDE in any machine we build either.

To pick the fastest disks, pay close attention to average seek and latency time. The former is an average time required to seek to any track; the latter is the maximum time required for any sector on a track to come under the heads.

Of these, average seek time is much more important. When you're running Linux, a one millisecond faster seek time can make a substantial difference in system throughput. The manufacturers themselves avoid running up seek time on larger-capacity drives by stacking platters vertically rather than increasing platter size. Thus, seek time, which is proportional to the platter radius and head-motion speed, tends to be constant across different capacities in the same product line. This is good because it means you don't have to worry about a capacity vs. speed trade-off.

One Disk or Two?

I always build with two disks: one system disk and one home disk. There are two good reasons to do this that have nothing to do with the extra capacity. One is the performance advantage of being able to interleave commands to different physical spindles. The other is I am quite a bit less likely to lose two disks at once than I am to trash a single one.

Let's suppose you have a fatal disk crash. If you have only one disk, good-bye Charlie. If you have two, maybe the crashed one was your system disk, in which case you can buy another and do a new Linux installation, knowing your personal files are safe. Or maybe it was your home disk; in that case, you can buy another home disk and restore it from backups (you did keep backups, right?).

Debian, Installing on Hardware Your Distribution Doesn't Support and the ReiserFS Two-Step

Easier Choices

Max out your memory. Lots of free memory will improve your virtual-memory performance. Fortunately, with RAM as cheap as it is now, a gigabyte or three is unlikely to bust your budget even if you're economizing.

You'll need a CD-ROM or DVD-ROM drive (you'll almost certainly be installing your Linux from it). You have a SCSI system, so get a SCSI CD-ROM. That's pretty

much the end of spec, as there are only a few models of SCSI DVD-ROM, and SCSI CD-ROMs are a generic item.

We'll want a good, high-volume backup device, too. Large disks are so cheap that backing up your home directory to another disk seems an attractive alternative, but it's still good to be able to make backups that you can separate from your system and store off-site, in case of disaster. We'll go with a DDS tape drive. Even if you're building on the cheap, the less expensive CD-ROM burners aren't a good idea for mass backup. The problem is the per-megabyte cost of the media, which you can't reuse. Rick adds: “Tape is also faster, more rugged both in storage and in the process of recording (jostling a DAT drive doesn't destroy the ongoing backup), doesn't require gobs of scratch space for assembling image files and is way, way, easier to automate.”

Speaking of faster, one of the things you want most in a tape drive is transfer speed. This is a good reason to go with the newer DDS4 tape drives, which have speed that is typically half of the older DDS3 drives.

Noise Control and Heat Dissipation

An increasingly critical aspect of machine design is handling the waste heat and acoustic noise of operation. Cooling is centrally important if you want your ULB to last because thermal stress from waste heat is almost certainly what will kill it. On the other hand, cooling makes acoustic noise, which human beings don't tolerate well. It's fair to say we've already reached the point at which the thermal load vs. cooling-noise trade-off is the effective limiting factor in the performance of personal machines.

So how do we manage this trade-off for a personal, desktop or desk-side machine? Being willing to pay a price premium for cool-running and low-noise parts can help a lot. Even clueful system integrators can't afford to do this because they're under constant competitive pressure to cut costs by using generic components. But, we aren't economizing here; we get to do it right.

The Recipe File

Now that we've laid out the principles, it's time to do the practice—specify and build a machine.

Processor, Motherboard and Memory

In July 2001, the clear standout choice for a ULB motherboard is the Tyan Thunder K7, model S2462 (see the Sidebar titled “AMD, SMP, AGP and LEDs: the Tyan Thunder K7 S2462”).

AMD, SMP, AGP and LEDs: the Tyan Thunder K7 S2462

There are good and bad consequences of having your peripherals onboard. The good ones are that the board has fewer points of failure and will throw less heat. The downside is that integration could make fault recovery more difficult. You want to minimize the chance that a failure in one onboard component will require an immediate motherboard swap. On the S2462, all the onboard peripherals can be jumpered out or disabled from the BIOS setup screens.

Choosing a Case

Internal expansion space isn't very important anymore because two-drive bays will hold more disk than you'll ever need. External bays are more important; you want one CD-ROM, one tape, one floppy and perhaps a DVD drive. That's one exposed floppy bay, three exposed half-height 5.25" bays and two internal bays.

There are three other important things you want from a case: good airflow design, component accessibility and noise attenuation, in that order. Finally, you may want your case to look neat. Good airflow design is actually the best reason to buy a large case. You want plenty of room for cool air to flow around the heat-generating electronics.

Tyan's site lists cases that have been qualified with the S2462, so I shopped around for a full tower on that list. Antec's Performance Series offers a number of cases that Tyan qualifies, and the swing-out side panel and quick-release drive bays featured on all of them appealed to me. When my design evolved to include a DVD player and the front-panel controls for a sound card, I went with the SX1200, the full-tower version with seven exposed bays.

Power Supply and Cooling

For the power supply, the three of us agreed on a vendor: PC Power & Cooling. PCP & C has a reputation for making good units and, as a bonus, quiet ones. PCP & C justified our confidence when they told me of their brand-new 450A4 unit, specifically designed for use with the S2462. And at 44dBA, the A4 counts as pretty quiet.

In May, Tom's Hardware compared 46 CPU coolers. The clear standout is the Silverado from Noise Control, Inc., rated best in cooling performance at 30°C and second-best in noise emission, only 37dBA. Typical coolers emit about 50dBA. The Silverado's only real drawback is that it's large—80mm long, 56mm wide, 113mm high—so you need to be careful about case clearances.

We can avoid having our case fans add more than a bare minimum to the machine's decibel output by specifying cooling fans that have ball bearings rather than the cheaper and more common sleeve bearings. This will cut machine noise by an appreciable degree, especially the annoying, whining high-frequency component, which is mostly bearing noise.

PC Power & Cooling makes 20dBA Silencer 80mm ball-bearing case fans. Specify the three-pin connectors to plug into the motherboard, not the four-pin connectors meant to be plugged into the power supply.

Mass Storage

We're going to be specifying fast-wide LVD drives, the cutting edge in SCSI devices. Within that class, the important statistics are seek time, rotational latency, capacity, heat dissipation and noise output. Mean time between failure is long enough on the leading brands that you're quite unlikely to see one before your system is years obsolete.

A StorageReview.com search confirmed anecdotal evidence from Rick Moen. He likes IBM's current product line, the UltraStar. With a 4.2ms seek time, they edge ahead of competition from Seagate, Quantum and Fujitsu. Rick believes they run relatively cool, too, and we hear they smoked the competition in some comparative trials run by Evi Nemeth at the CAIDA Project. So we'll add two IBM UltraStar 36Z15 drives to the parts list.

We also want to be able to read (and write) CD-ROMs. Again, StorageReview.com confirms Rick's anecdotal report, tapping the 32-speed Plextor PX W1210TS as the best-of-breed among SCSI CD-RW drives.

Rick observes:

> CD-R/CD-RW drives by their nature have head assemblies much more massive than those of ordinary read-only CD drives. Why? Because they mount burn lasers. Much greater mass means much greater inertia and much faster mechanical wear, and the considerable heat generated during burn cycles also takes its toll. Accordingly, the MTBF times for CD-R/CD-RW drives are markedly shorter than for regular CD drives. One should not use CD-R/CD-RW drives for mundane read operations, but rather only for CD-burning. Accordingly, if you really have the need for a CD-R or CD-RW drive, you also need a second, read-only drive for everyday CD-reading.

Daryll Strauss chimes in with: “Buy a DVD-ROM rather than an ordinary CD-ROM. Typically the transfer rates are just as good, if not better, because the base DVD rotational speed is higher to begin with.”

A DVD is a must-have for another reason; any true dream system for a Linux hacker must include the ability to violate the anti-fair-use clauses of the Digital Millennium Copyright Act by playing DVDs, even if (like me) the hacker is basically uninterested in DVDs per se. It's ethically imperative.

Presently, only two models of SCSI DVD-ROM are available: the 304S/305S by Pioneer and the SD-M1201B by Toshiba. The Toshiba is 5X as a DVD drive and 32X as a CD-ROM drive; the Pioneer's numbers are 10X and 40X. Easy call, especially since the Toshiba is actually more expensive.

History says that the top-of-the-line Hewlett-Packard tape drive is either going to be the best-of-breed or close. The top-of-the-line HP DDS4 drive appears to be the C5685, with a capacity of 40GB and a transfer rate of 21.6GB/hour (assuming hardware compression).

Monitor Graphics Card and Sound

For my purposes, clearly displaying a lot of text at relatively small font sizes is the most important thing I want a monitor to do. Thus I pick the only monitor *PC World* rates as excellent at both text and graphics, the Mitsubishi Diamond Pro 2060u. It supports 2048 × 1536 at 75Hz, a refresh speed comfortably above flicker level.

Daryll is a graphics expert and part of the team working on the Linux drivers for ATI's high-end Radeon card. He tells us that for the foreseeable future (or at least until NVIDIA gets a clue about open source) the Radeon will be the best high-end graphics card with entirely open-source drivers. So we add one ATI Radeon 64MB card.

Because this is a development box rather than a gaming machine, it's more important that a sound card be well supported with stable drivers than that it hug the bleeding edge of audio technology. The safe choice seems to be the SoundBlaster Live Platinum 5.1. ConsumerSearch's top speaker pick, rated excellent for both game play and music, is the Klipsch ProMedia 2.1.

Graphics: Safety or Speed?

opensource.creative.com: SB Live

Oh, for a Real Keyboard!

Like many hackers of a certain age, I imprinted on the IBM Model M keyboard about 20 years ago. They have a relatively stiff travel with a sharp break and a positive keyclick that can only be described as crunchy. They inspire cult-like devotion. It's still possible to buy the real Model M, armor-plated case and all.

They're not being manufactured anymore, but old stocks are still being sold. You want these IBM model numbers: 42H1292 (IBM 101-key, buckling-spring keyboard) and 1393278 (IBM SpaceSaver compact, heavy-duty 84 keyboard). They're both available from Unicomp. The dream system will get one of the 101-key PC-2 versions.

Keyboard

Miscellanea

For my own use, I'll keep my original three-button Logitech TrackMan Marble. Sadly, Logitech doesn't make the original Marble any more; the replacement has a rather obtrusive wheel replacing the middle button.

There is only one possible modem for the dream system: the US Robotics V.Everything, external version. This featureful, rock-solid, reliable modem is the first choice of discriminating hackers everywhere. Rick has written an entertaining rant on the likely consequences of choosing lesser external modems, or any internal modem at all.

The floppy drive is a relic of the age before bootable CD-ROMs. Occasionally you'll want one for booting up diagnostic software. A plain old TEAC 1.44 3.5" drive will do.

Oh, yes, the software. I realize that the topic of favorite Linux distribution is a religious war, but I can't resist putting in a plug for my own favorite: KRUD Linux from Kevin Fenzi and the good folks at tummy.com. Subscribing to KRUD gives you a Red Hat base plus a monthly update, including all security fixes and a tasty selection of additional programs and tools.

System Integration

We have two SCSI controllers. That's good, because we also have both LVD and single-ended SCSI devices in our parts list. Daryll observes:

> LVD drives can drive the bus at 40MHz and 80MHz, whereas single-ended cannot. If you mix single-ended and LVD, the bus degrades to single-ended. So a bus with a single-ended device tops out at 20MHz Wide SCSI or 40MB/s, whereas LVD gets you up to 160MB/s.

Thus, we want to assemble the dream machine with two SCSI chains: a high-speed wide/LVD chain for the hard drives and tape, and a low-speed narrow/single-end chain for the CD-RW and DVD-ROM. We used an SM-20 from The Mate Company to convert the second motherboard channel to 50-pin narrow SCSI.

Because the hard drives are likely to be significant heat generators, we mount them with the spare internal bay between them, rather than stacking them in adjacent drive bays, to get better airflow.

The Antec case makes it possible to mount the intake fan directly in front of the hard disks. Normally, with drives in this class, the drives and the bay enclosure become uncomfortably hot to the touch; with this setup, the warmth is barely noticeable. This is a good thing because it probably extends the expected lifetime of the drives significantly. Another fan near the power supply at the rear helps pull air out of the machine. We ended up mounting a third fan because we noticed the memory chips seemed to be running hot.

We'll have two expansion cards in the machine, the SoundBlaster Live! and the Radeon. The Radeon will probably tend to run hot, the SoundBlaster not. Happily, the Radeon lives in the AGP slot at the upper end of the slot row, where the air it heats will be sucked into the two rear fans.

How does our noise budget look? IBM says our UltraStars emit 48dBA each, PCP & C says the power supply emits 44dBA and the fans 20dBA each, and Tom's Hardware rated the Silverado at 37dBA (but there are two). Applying the logarithmic-sum formula gives us 52dBA as the level of interior noise. Assuming the case blocks 8dB, that will leave us with an exterior noise level of 44dBA adjacent to the case. We can trim another 5dB or so by putting the machine desk-side.

Recalculating with four or five case fans barely nudges the second decimal place in the total. This means that in case our initial burn-in reveals a heat problem; we've got room to cool things down without making the machine substantially noisier.

Building the Machine

Gary Sandine and John Pearson at Los Alamos Computers undertook to assemble my Ultimate Linux Box; in fact, they assembled two, one for me and one for Linus Torvalds. They solicited the vendors on our list for donations of parts, and their courage was rewarded when IBM generously volunteered $15,000 for the project budget.

Troubleshooting

See “Troubleshooting the Ultimate Linux Box” on the *Linux Journal* web site.

Conclusion

I find it impressive that, after having specified it on a cost-is-no-object basis, the total system cost is so low. I tried to gold-plate as much of the system as possible and load on all the extras and accessories I could, and I was still unable to raise the total parts bill over $7,000 US.

If we discarded the most extravagant peripherals—the Klipsch speakers, the Radeon, and the DVD and DDS drives—the cost would drop to a quite reasonable $4,200 US or so. As Rick pungently observes, “People pay more than that for crap computers every day.” This design will be available for purchase from Los Alamos Computers as the ULB-200108.

And how fast does it build kernels? After **make clean**, the Ultimate Linux Box builds the ULB's 2.4.8 Linux kernel from a cold standing start (**make -j3 'MAKE=make -j3' dep; make -j3 MAKE=make -j3' bzImage**) in 2 minutes and 21 seconds flat. Sweeeet.

Resources

email: esr@thyrsus.com

Eric S. Raymond is a wandering anthropologist and troublemaking philosopher who happened to be in the right place at the right time and has been wondering whether he should regret it ever since.

Archive Index Issue Table of Contents

Advanced search

2001 Readers' Choice Awards

Heather Mead

Issue #91, November 2001

See how your preferences compare with the rest of *LJ*'s intrepid readers, all of whom are highly attractive.

Sorting through the results of the 2001 Readers' Choice Awards, ballots gathered from six weeks of on-line voting, it's clear that *Linux Journal* readers have opinions on everything—lots of opinions. Also evident is the fact that more options, tools and methods exist now than ever before; a good thing to keep in mind during a year so turbulent.

This year over 6,500 readers voted in 24 categories, from favorite Linux book and office suite, to favorite communications board and backup utility. Thank you to everyone who participated. Now, on to the results.

Favorite Distribution

1. Red Hat 2. Debian 3. Mandrake

Red Hat received 30% of the votes this year, repeating their victory from last year. Debian moved from fourth to second place this year, and Mandrake stayed at third. Linux from Scratch and the Polish Linux Distribution (PLD) were the most popular write-ins.

Favorite Graphics Program

1. The GIMP 2. xv/xview 3. CorelDRAW

The winner here is what you would expect; the GIMP received 77% of the total votes. xFig and Photoshop were the favorites among write-in votes, with pleas for a Photoshop port abundant.

Favorite Word Processor

1. StarOffice 2. AbiWord 3. Kword

StarOffice wins this category for the second consecutive year. Its nearest competitor, AbiWord, received half as many votes, and the spread between it, KWord and Emacs was only 19 votes. Though not really a word processor, LaTeX was the favorite write-in.

Favorite Text Editor

1. vim 2. vi (and clones) 3. GNU Emacs

We took your advice from last year and split vi and vim into separate categories. This time around vim wins with twice as many votes. **mcedit** took the write-in vote.

Favorite Desktop Environment

1. KDE 2. GNOME 3. Window Maker

This was one of the most popular categories, and KDE is the clear winner, receiving 40% of all votes. GNOME came in second with 24.5%, and the favorite write-in was xfcr. And special mention, of course, for the command line.

Favorite Office Suite

1. StarOffice 2. KOffice 3. WordPerfect

StarOffice is your favorite word processor and your favorite office suite this year, same as in 2000. KOffice finished a strong second this year, after a small showing last year. And, the write-in opinions want everyone to check out Open Office.

Favorite Programming Language

1. C 2. Perl 3. C++

Here's another category where we took your advice from last year and split C/C++ into separate categories because, hey, they're not the same. Java and PHP finish out the top five, with Python just missing out by 15 votes. Kylix/Object Pascal had a strong write-in showing, over 200 votes.

Favorite Development Tool

1. GCC 2. Emacs 3. KDevelop

GCC took first place again this year, but by a percentage significantly lower than last year. Emacs continues to prove its flexibility here, too. Last year's popular write-in, KDevelop, took third place this year, while Borland's Kylix made another strong write-in showing.

Favorite Shell

1. bash 2. tcsh 3. ksh

Eighty-one percent of all voters chose bash as their favorite shell, with tcsh coming in a distant second. **ksh** came in third, but only by receiving five more votes than fourth-place zsh.

Favorite Processor Architecture

1. AMD Athlon 2. Intel Pentium 3. PowerPC

Readers' favorite processor is the AMD Athlon, tallying 42% of the votes. AMD's Duron, K6-II and Celeron were popular write-ins. Quite a few votes commented that their selection of Pentium was out of necessity, not performance.

Favorite Communications Board

1. Cyclades 2. Digi International 3. Equinox

This one received the fewest total number of votes and, judging from some of the comments, it's because not everyone knows what we're talking about. Well, it's not a surfboard and it's not a bulletin board. Of those who did get our meaning, Cyclades is the favorite.

Favorite Database

1. MySQL 2. PostgreSQL 3. Oracle

For a second year in a row, MySQL beat PostgreSQL by a 2:1 ratio. Combined, they received almost 80% of all votes. Write-in favorites were the Red Hat Database (powered by PostgreSQL) and GemStone/S.

Favorite Backup Utility

1. tar 2. Amanda 3. Arkeia

tar is by far the favorite backup tool our readers use. Amanda and Arkeia came in second and third place, respectively, but only by a one-vote difference. BRU, dump and homemade backup tools are also still in widespread use.

Favorite *Linux Journal* Column

1. Cooking with Linux 2. Kernel Korner 3. At the Forge

Fans of Marcel Gagné, author of the Cooking with Linux column, came out in full force this year and moved him to the top position. Second was the revolving-author Kernel Korner. The most popular write-in—no, we're not making this up—was “all of them”.

Favorite Programming Beverage

1. Coffee 2. Water 3. Tea

Who would have thought this would be one of the most debated categories on the ballot? We added Mountain Dew after last year's outcry, only to be chided for forgetting Dr. Pepper (Other Soft Drinks). Coffee keeps its power over all of us, and over 100 brave souls admit to consuming foo-foo frilly coffee drinks.

Favorite Linux Game

1. *Quake III* 2. *xBill* 3. *Tux Racer*

Some write-in comments claim games aren't for serious Linux users, but they sure do bring in the money and drive development. And we all need a little fun now and then, right? *Quake III* is the favorite again this year, and the classics *Mahjongg* and *Shisen-sho* dominate the write-ins.

Favorite Web Browser

1. Netscape 2. Mozilla 3. Konqueror

Netscape captured 30% of the votes this year; Mozilla (bugs and all) trailed by just over 300 votes. Internet Explorer was the favorite write-in, but a sentiment shared by many is that they “all suck, just differently”.

Favorite Linux Web Site

1. Slashdot 2. Freshmeat 3. LinuxToday

From company web sites promoting Linux products and service, to international sites and community help sites, the list of write-in favorites goes on and on. Of course, Slashdot is still the first stop for the majority of voters, almost 30%. The most popular write-in is www.linuxnews.pl, a Polish Linux site.

Favorite E-mail Client

1. Netscape 2. KMail 3. pine

Although there are winners in this category, none claim dominance. Barely 70 votes separated Netscape, in first place, from third-place mutt. One-time favorite, elm, has fallen to the bottom on the list.

Favorite Instant-Messaging Client

1. Xchat 2. Jabber 3. BitchX

For those that participate in IM, Xchat is the favorite over Jabber by 4%. The most popular write-ins are Licq and GnomeICU. And quite a few you express disdain for all forms of IM or IRC. Even more of you resort to AOL, Yahoo or MS's versions 'cause “that's what all my friends use”.

Favorite Distributed File-Sharing System

1. Gnutella 2. Freenet 3. OpenNAP

Remember last year when debate over this whole topic was being played out in every coffee shop and courthouse across the US? Well Metallica avoided the poorhouse, and Gnutella and Freenet remain the preferred methods of file sharing. Among the write-ins, audiogalaxy and MORPHEUS are mentioned most.

Most Indispensable Linux Book

1. *Linux in a Nutshell* by Ellen Siever 2. *Running Linux* by Matt Welsh, et. al. 3. *Linux System Administration* by Vicki Stanfield, et. al.

Perennial favorites, *Linux in a Nutshell* and *Running Linux*, once again find themselves in the top two spots, but in a reversal from last year, *Linux in a Nutshell* is this year's first choice. Man pages and other on-line documentation are the write-in favorites, and it looks like just about every other Linux book printed received at least one vote.

Favorite Ad Filtering Tool

1. Junkbuster 2. Homemade 3. Squidguard

One person wrote in, “Ads are a part of the page's design and intended viewing.” Apparently, he or she is alone in that sentiment, as just about everyone else uses a filter, Junkbuster being the one most named. A lot of people are making their own proxies, too.

Favorite Audio Tool

1. xmms 2. pmg123 3. RealAudio

Receiving 58.4% of the total vote, xmms is once again readers' favorite audio tool. Comparatively, none of the others accumulated more than 9%. Grip shows up most amongst the write-ins, and support is still strong for the ogg tools.

Heather Mead is associate editor of *Linux Journal*. In her free time, she enjoys photography, movies, music and writing short stories.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Data Modeling with DODS

Reuven M. Lerner

Issue #91, November 2001

Part of the Enhydra server, DODS tries to bridge object andrelational databases.

Relational databases form the backbone of most serious web applications because they make data storage and retrieval easy, safe and flexible. This setup normally works wonderfully, until developers begin to write their programs using objects, which have a completely different paradigm. Is there any way for us to close the gap between the object and relational worlds?

Actually, many ways exist to map relational databases to objects and methods, and most programmers have found themselves writing such systems on their own. As we saw last month, Perl programmers can get some assistance from Alzabo, a module that helps programmers to create tables and then provides a method-based interface to manipulate them.

This month, we will look at DODS (Data Object Design Studio), a tool similar to Alzabo in spirit, except that it is aimed at Java programmers. DODS is a central part of Enhydra, whose upcoming version (Enhydra Enterprise) is expected to be the first open-source application server to implement J2EE (Java 2, Enterprise Edition).

Right now, Enhydra Enterprise is still in beta testing, and while the DODS support appears to have improved since the first versions, I was told by David Young, the Enhydra evangelist at Lutris, that DODS from Enhydra 3.x is more stable. So that I could try things more easily, Lutris sent me a copy of EAS (Enhydra Application Service), their commercially supported and enhanced version of Enhydra.

I'm not entirely sure what the differences are between EAS and the open-source Enhydra server; enhydra.org indicates that EAS is “based on” Enhydra, but the differences between EAS and buying a copy of Enhydra are not at all

obvious. I will assume that the copy of EAS I installed is largely similar to Enhydra 3.x, but this might not be an accurate assumption.

DODS Overview

DODS, like the Perl Alzabo system we discussed last month, has two goals: to help design databases with a high-end tool and to provide a set of objects and methods for working with that database. While Alzabo is a server-side web application, DODS is a client-side application written in Java that lets you define and edit your database definitions.

The goal of DODS is to create parallel SQL definitions and Java classes that describe the same relational database. You then feed the SQL definitions into your database and use the Java classes to access them.

Moreover, DODS is designed to work with different databases; at present, it works with PostgreSQL, MySQL, Sybase and Oracle, but it may work with more in the future. Since the actual SQL queries are written by an object middleware layer, this means you can move your Enhydra programs from one database to another without having to rewrite them. In reality, of course, things are more complicated. For example, Enhydra's support of PostgreSQL is not very impressive, ignoring the SERIAL data type (which is really just a sequence) and unaware of referential integrity constrations, such as foreign keys. Nevertheless, the goal is an admirable one, and I'm looking forward to seeing how Enhydra 4.x handles this. As time goes on, I expect that DODS will continue to improve its support for different databases, creating appropriate queries for individual SQL dialects.

Creating the XMLC Document

Let's create a simple web/database application using Enhydra and DODS to demonstrate how it will work. I'll use PostgreSQL in my examples, both because it's an excellent open-source database and because DODS supports it. Our example, as has been the case for the last few months, will be a simple weblog (or blog) application that displays entries in a database table in reverse chronological order. Writing such a program is not particularly difficult, making it all the more attractive as a test of DODS and Enhydra.

Our first stop will be the Enhydra appwizard, which creates the directories and skeleton files for our basic Enhydra application. The appwizard is in *$ENHYDRA*/bin, where *ENHYDRA* is an environment variable naming the location of your Enhydra installation. (When I installed it from CD using RPMs for my Red Hat Linux box, ENHYDRA was set to /usr/local/lutris-enhydra3.5.2.)

On the first appwizard screen, I was allowed to choose between a standard web application and an Enhydra super servlet. I chose the latter. On the next screen, I chose an HTML project (rather than a wireless WML project), named the project “blog” and put it in the il.co.lerner package class. I also accepted the default home for Enhydra applications, ~/enhydraApps/. I chose not to associate a copyright with my source code and then clicked Finish, which created 18 new files in ~/enhydraApps/blog.

Now that I've created the skeleton for my application, I'll modify the default Welcome page that comes with Enhydra. We will have to do this in two parts; first, we'll modify the HTML file Welcome.html, which my computer placed in ~/enhydraApps/blog/src/il/co/lerner/presentation/Welcome.html.

Note that this file consists not just of HTML, but of tags that will be processed by XMLC (see At the Forge in the August 2001 issue of *Linux Journal* to see just what that means). We'll change it to display the latest information from our weblog, rather than the standard page, as you can see in Listing 1. The only difference between an XMLC document and a standard page of HTML is we mark the parts we want to modify inside of tags, with an id attribute. For example:

```
<p><b><span id="date">Date</span><b></p>
<p><span id="text">Text</span></p>
```

Listing 1. Welcome.html, the XMLC Input File We'll Use to Display Our Weblog

If we display this file literally in our web browser, we'll see the words Date and Text. But users will not retrieve this HTML document directly. Rather, XMLC will turn our document into a Java class. We will then use the WelcomePresentation class to create an instance of the document, using automatically generated methods to modify the date and text sections.

Using DODS

WelcomePresentation, however, will fill in the date and text sections with information that it retrieved from a relational database table. We will have to create the table, as well as populate it with some information, before we can continue.

This is where DODS enters the picture. The DODS program, which is in $ENHYDRA/bin/dods, is another graphical client-side program written in Java. When working with DODS, we have to remember that we're using a bridge between two different paradigms, and the terminology may often be a bit weird.

DODS begins with a package, which is a container for all of the tables and other attributes that we will create. By default, as you can see from the initial DODS window, the package is named root; I changed this to blog by clicking on the root folder and selecting Package from the Edit menu.

We will create a database table (BlogEntries) with two attributes: the date and the contents. (These match the two id tags that are in our version of Welcome.html.) We first add a new table to BlogEntries, using the Insert menu, choosing “data object” and giving it the name BlogEntries.

Next, we must add two fields (date and text) to our table. In order to do this, click on the BlogEntries name in the upper left-hand corner of the DODS window, and use the Insert menu to add each new attribute. Both of our attributes will be of type “varchar”, meaning that we'll treat them both as text. And while this will work perfectly well for our purposes, it's unfortunate that DODS doesn't support the PostgreSQL TIMESTAMP data type, a tool that allows us to perform all sorts of clever and sophisticated manipulations on dates and times. So we'll store the dates in text-only format, knowing that we can use ORDER BY to retrieve them in order—and not much more than that.

Because we want to ensure maximum speed for our web/database application, and because we'll be inserting text much less frequently than we retrieve it, we will tell DODS to make both of our fields indexable and queryable. The former will modify the SQL defintion, ensuring that an index is created for the attribute. The latter will create additional methods that allow us to retrieve information based on this column.

Finally, we choose PostgreSQL from the Database menu, telling DODS to create PostgreSQL-style SQL, rather than any other database.

Once we're done creating our table with DODS, we export it into DOML (using File Save as), an XML format that describes our tables and can be used to generate both the SQL and Java. Make sure to save the DOML file into the source directory within your project and package; thus, I put it in blog/src/il/co/lerner. Once our DOML file is complete (see Listing 2), we can turn it into Java and SQL with the **build all** command from the File menu. Building everything results in the creation of a number of files within the data directory, so when you're asked where you want to build things, select the data directory in which you stuck the blog.doml file. A window will pop up, describing what DODS is doing at each moment. If everything is successful, you can exit from DODS.

Listing 2. blog.doml, the Automatically Generated File that DODS Creates

Creating the Database

Running **build all** from DODS turns the DOML file into a set of files under the data directory. This directory now contains not just a Makefile (previously empty), but also the blog subdirectory in which the following four Java classes now exist: BlogEntriesDO, a data-object class similar to Enhydra's presentation objects for displaying information; BlogEntriesDataStruct, which actually contains the data from our table; BlogEntriesDOI, an interface for the BlogEntriesDO class; and BlogEntriesQuery, which allows us to query fields we've described as queryable.

In addition to the Java source code that was created, we also have several files containing SQL. In particular, the data directory contains create_tables.sql, which we can use as input to create our database.

To do this, we use the CREATEDB command, which is normally executed by an authorized PostgreSQL user from the UNIX shell. (This does not necessarily mean the root user; check your PostgreSQL installation documentation to see how to create PostgreSQL superusers.)

We thus say

```
CREATEDB blog
```

We can then query that database interactively with

```
psql blog
```

To create our tables using the automatically generated DODS script, we use the **\i** command from within psql:

```
\i /home/reuven/enhydraApps/blog/src/il/co/lerner/ data/create_tables.sql
```

You should see several CREATE messages, and then the psql prompt once again. Using the **\d** command, we see that DODS actually didn't create a table named BlogEntries. Rather, it created two tables, one named objectid and the other (primary) table named newdbtable, which is the default. The objectid table comes in place of a PostgreSQL sequence, demonstrating one limitation of this sort of generic tool, and has a column “next” that indicates which ID comes next. We thus insert information into the newdbtable table, adding a new row to objectid each time we do so.

Let's add several items to our table so that we'll then be able to retrieve them:

```
INSERT INTO newdbtable (entrydate, text, objectid, objectversion)
VALUES (NOW(), 'First blog entry', 1, 1);
INSERT INTO objectid ( next ) VALUES ( '2' );
INSERT INTO newdbtable (entrydate, text, objectid, objectversion)
```

```
  VALUES (NOW(), 'Second blog entry', 2, 1);
  INSERT INTO objectid ( next ) VALUES ( '3' );
```

Now that we have two blog entries inserted, we can exit psql and go onto modifying our Java class.

Pulling It All Together

Our WelcomePresentation.java class is where most of the magic happens. It creates instances of both Welcome.html and of our database objects, retrieves the query results and then inserts those results into the HTML file.

After you modify WelcomePresentation.java, as shown in Listing 3 [available at ftp.linuxjournal.com/pub/lj/listings/issue91/5426.tgz], run a **make** from the top-level directory for this project. Enhydra will compile your Java classes, double-check that everything is where it should be and get your application ready to run.

Notice in Listing 3 that we had to modify the definition of the “run” method such that it returns two new exceptions: NonUniqueQueryException and DataObjectException. These are generated by the various data objects that we've created, and since we aren't going to catch these exceptions, we must indicate to the caller that we may raise them.

Listing 3 uses the Enhydra QueryBuilder to create an SQL query using methods created by DODS. We first create an instance of BlogEntriesQuery, one of the automatically created classes:

```
  BlogEntriesQuery blogq = new BlogEntriesQuery();
```

We want to retrieve all rows until now, in reverse order by entryDate:

```
  blogq.setQueryEntrydate("NOW()", QueryBuilder.LESS_THAN);
  blogq.addOrderByEntrydate(false);
```

There are also methods for adding WHERE clauses to our SQL query, letting us create arbitrarily complex SQL queries.

Finally, we retrieve an array of matching rows, each of which is represented by a BlogEntriesDO object:

```
  BlogEntriesDO[] blogEntries = blogq.getDOArray();
```

We're only going to display the most recent one, so we will simply get the first element of our array. We use the method in our “welcome” object, created by XMLC, to insert the appropriate text in our document:

```
welcome.setTextDate(blogEntries[0].getEntrydate());
welcome.setTextText(blogEntries[0].getText());
```

Once we have modified WelcomePresentation.java, we create the application by running **make** from our top-level project directory. If you see any errors in your Java program, you can correct them and rerun make as often as you want.

Theoretically, you could now run the application by changing into the output subdirectory and running **./start**. But our application will fail if we do this, since it doesn't yet know where to look for the PostgreSQL .jar file. In addition, it's useful to get full debugging output from Enhydra (or any application) when we're first using it, so that we can identify and fix problems more quickly.

We must modify three files in order to get things to work. First, we need to modify $ENHYDRA/bin/multiserver by adding a reference to the PostgreSQL JDBC driver's .jar file. To do this, we simply modify the multiserver (which is a shell script that invokes a Java program), changing the lines under the comment “build up classpath” to the following:

```
# Where is the PostgreSQL JDBC .jar file?
PG_JDBC=/usr/share/pgsql/jdbc7.1-1.2.jar
if [ "X${CLASSPATH}" = "X" ] ; then
    NEWCP=${ENHYDRA_CLASSES}${PS}${PG_JDBC}
else
    NEWCP=${ENHYDRA_CLASSES}${PS}${CLASSPATH}${PS} ${PG_JDBC}
fi
```

Next, we modify blog.conf. Every Enhydra project has a configuration file that tells the system which database to use, as well as a number of other properties. In my particular case, the configuration file is blog/output/conf/blog.conf and consists of a lot of name-value pairs for my application.

We must modify several parts of the “Database manager” section in order to point to our programs. You can see the full section, as it needs to be, in Listing 4 [available at ftp.linuxjournal.com/pub/lj/listings/issue91/5426.tgz].

Finally, we modify servlet.conf. Although this doesn't need to be modified, I find it useful to turn on DEBUG mode by modifying the following two definitions:

```
Server.LogToFile[] = EMERGENCY, ALERT, CRITICAL, ERROR, WARNING, INFO,
DEBUG
Server.LogToStderr[] = EMERGENCY, ALERT, CRITICAL, ERROR, WARNING, INFO, DEBUG
```

The most important thing to realize about blog.conf and servlet.conf is that they are regenerated every time you do a top-level make. So once you have modified them in this way, never do a top-level make again. You will be quite sorry (as I have been) if you do so. Rather, do a **make** from within the presentation directory.

Once you have modified the configuration in this way, you can go into ~/enhydraApps/blog/output and run **./start**. You will see the server start up, plus a fair amount of debugging information if you activated DEBUG in servlet.conf and logging in blog.conf.

You can check out your creation by pointing your web browser to point 9000, the default port for Enhydra applications: http://localhost:9000/. If all is well, you should see the output from our weblog in your web browser.

Conclusion

DODS is better than Alzabo at providing a nice object layer above the SQL. Moreover, it seems to provide a better and more stable set of methods for creating queries and working with their results. However, DODS suffers from several of the same problems as Alzabo and other relational-object mapping tools. Some problems include having to learn a new way to execute all of those SQL queries you've been working with for years and complex queries that can be frustrating to write, since you have to rephrase them as method calls. Also, the generic nature of a tool like DODS means that you won't get the specific benefits of your favorite database. In the case of PostgreSQL, DODS didn't seem to take foreign keys or sequences into account, both of which would have made for more solid table designs.

DODS works well when coupled with the rest of Enhydra, though. As with XMLC and super servlets, I found DODS to be somewhat overwhelming and clunky at first, and then useful and clever. As a first stab at things, DODS in Enhydra is a good attempt to bridge the gap between the object and relational worlds. I look forward to the final version of Enhydra Enterprise, which will undoubtedly push things ahead even further.

Next month, we'll look at an increasingly standard way to not only bridge the relational and object worlds, but also to add transactional capabilities to our server-side Java applications. Enterprise JavaBeans represent services and data for web applications and have become increasingly popular among web developers who want to be able to find objects, work with them and store them to a database, without having to think too hard.

Resources

email: reuven@lerner.co.il

Reuven M. Lerner (reuven@lerner.co.il) owns a small consulting firm specializing in web and internet technologies. He lives with his wife Shira and daughter Atara Margalit in Modi'in, Israel. You also can reach him on the ATF home page, www.lerner.co.il/atf.

LINUX JOURNAL

Enterprise—Help for Networks

Marcel Gagné

Issue #91, November 2001

nPULSE and NetSaint, two web-based monitors for enterpriseenvironments.

When I look at the focus of this month's issue, François, my first thought is certainly the same as yours, *mon ami*. For me, the word enterprise summons the image of a certain starship searching space for new and exciting aliens, not to mention their local wines, *non*? But in the business world where many of our guests work, enterprise means *très grand*, as in big business.

Quoi? *Moi aussi*, François. When I think of a large office, or worse, many large offices, I think of the poor system administrators who must take care of all those computers, hubs, switches and PCs. My heart goes out to them.

Ah, François, what are you doing standing there? Our guests have arrived. Away to the cellar and bring up the 1996 Stellenbosch *immédiatement*. *Vite*! *Vite*! *Mes amis*, please sit down. Your table is waiting and François is fetching the wine. I envy you, *mes amis*, a bold South African red with a wonderfully long finish. I may even join you.

When you arrived, we were discussing the special pressures that working in a large enterprise puts on a system administrator. Imagine having to keep track of numerous systems spread across a distributed network, *mon ami*. People do not want to hear that the mail server is off-line; they want the administrators to know before them and have the problem already solved. It seems impossible. Add to that paper shortages in the supply cabinet and our friends have their hands full, *non*? That is why I have dedicated tonight's menu to helping them with this impossible job. Relief, *mes amis*, is on the way with a couple of wonderful recipes from great open-source kitchens.

The first item on tonight's menu is a little something designed to take the pulse of your network. What better name for such a tool than nPULSE.

In order to use nPULSE, you are going to need a couple of Perl modules (in fact, nPULSE is written in Perl) and my favorite port scanner, nmap. You can pick up nmap from www.insecure.org/nmap/index.html. You may also find it on your distribution CD, so look there first. While nmap is but one ingredient in the creation of nPULSE, I invite you to try it on its own. The latest version even comes with a nice GUI front end called nmapfe.

We also need a couple of Perl modules, *mes amis*, CGI.pm and Mail::Mailer (which is part of the MailTools package). There are two ways to install these modules. One is to go directly to the CPAN site and download both. You'll find the two modules at ftp.cpan.org, specifically in the following directories: /pub/mirrors/ftp.cpan.org/pub/CPAN/modules/by-module/CGI and /pub/mirrors/ftp.cpan.org/pub/CPAN/modules/by-module/Mail.

Installing the Perl modules is simple. The format is always the same, so let me show you with the MailTools package:

```
tar -xzvf MailTools-1.16.tar.gz
cd MailTools-1.16
perl Makefile.PL
make
make install
```

Now, do the same with the CGI.pm package, and you are ready to install the rest of nPULSE. *Mais oui*, I did say that there was another way to install Perl modules, so we shall look at that briefly. You can have the whole process happen magically by using the following commands:

```
perl -MCPAN -e 'install Mail::Mailer'
perl -MCPAN -e 'install CGI'
```

If you have never installed a Perl module in this way, be warned—the process can take awhile the first time around. You will have to go through a question-and-answer session asking where various files are located and what mirrors you wish to use for your downloads. The process also may install a fair number of packages so that you can do seamless Perl module installs in the future.

There you have it, *non*? Our modules are installed, we have nmap and are ready to continue with our nPULSE installation:

```
tar -xzvf npulse-0.54.tar.gz
cd npluse-0.54
./setup.sh
```

You'll be asked where you want nPULSE installed (the default is /usr/local/npulse) and a handful of other questions, including one about your personal architecture. Because nPULSE is multiplatform, you need to specify what you are running it on. I chose Red Hat 7 when I did this, but you'll see support there

for every major Linux distribution, as well as for SCO, Solaris, SGI and number of others. The final set of questions asks you to choose an administrator login and password for security authentication. Once that is done, nPULSE's web server is started on port 19000 (although you can change that as well). Point your favorite browser to http://your_server:19000/ to connect to the server.

The first thing you will want to do is configure the hosts being monitored. You do this with the Setup button on the page. When the new page appears, type an IP address or a range of addresses into the Discover box on the top right-hand side. Then click Go. **nPULSE** will scan that machine for available services, and these will appear in the large window below. Click Save and then switch to the Status screen where you can do a ping or port sweep. It is possible to switch from one view to another by choosing the various options in that little drop-down box beside the on/off switch at the top, right-hand side.

nPULSE is quite configurable with a number of possible responses to a particular host being on- or off-line. You can set it to send e-mails, page you or simply refresh the screen automatically for a bird's eye view of various networked systems. For a look at nPULSE in action, see Figure 1, where we zoom in on a PC client on the network. You'll notice that it is running a primitive operating system.

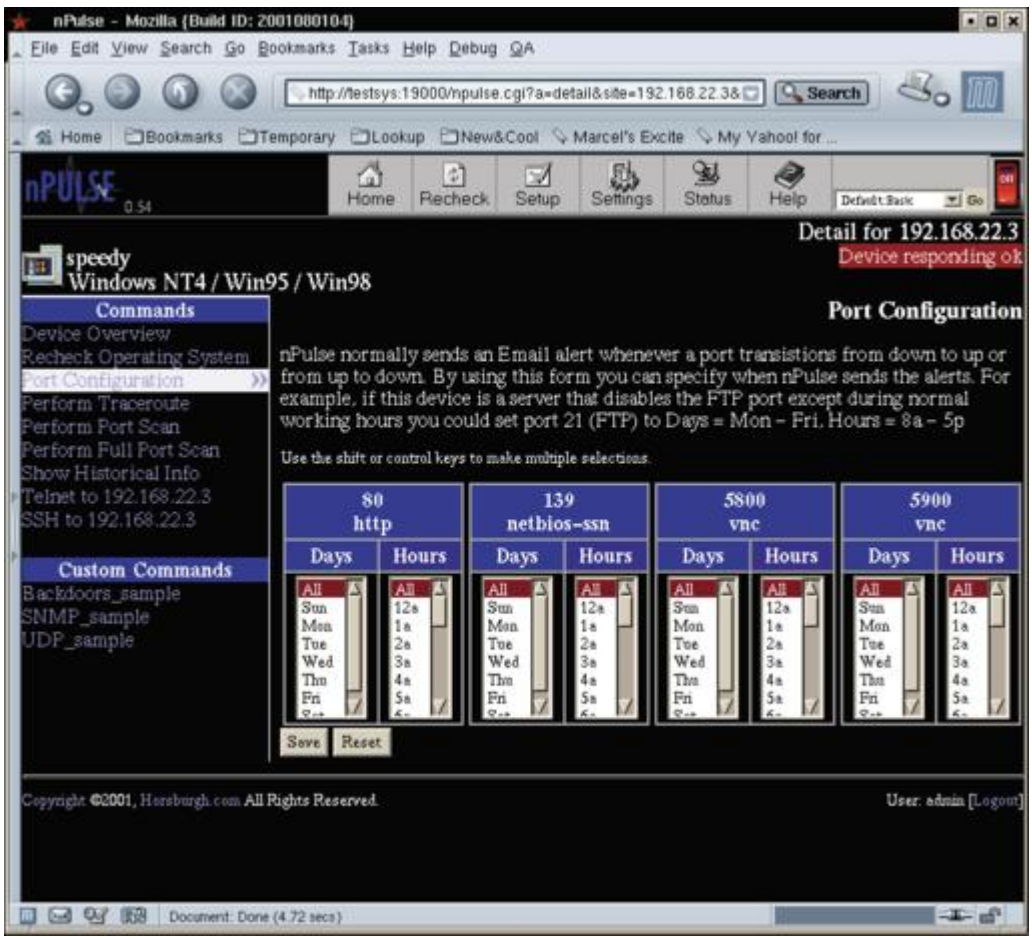


Figure 1. Zoom View of nPULSE at Work

The installation procedure automatically starts the server. To restart nPULSE, use

```
/usr/local/npulse/etc/restart
```

To stop nPULSE, use

```
/usr/local/npulse/etc/stop
```

The next item on tonight's menu is a saintly program called, quite appropriately, NetSaint. NetSaint is also a web-based network monitor that uses Apache to serve its web pages. Luckily, most Linux distributions come with the Apache server on the disk. You may already have it installed.

With NetSaint, you can keep track of your various networked machines, keeping an eye on all your important protocols, like SMTP, POP3, HTTP and so on. The program has built-in event notification similar to nPULSE and will send you an e-mail or a page in the event of a system going down. You even can define program responses to common problems so that things more or less take care of themselves. NetSaint also provides detailed logging of network status and events. Best of all, NetSaint is freely distributed under the GPL.

To get your copy of NetSaint, start by visiting the project site, www.netsaint.org. Then, extract and build the package as follows. Be warned, *mes amis*, there are a few interesting differences in the standard build when it comes to NetSaint. It pays to pay attention. The beginning is a familiar one:

```
tar -xzvf netsaint-0.0.6.tar.gz
cd netsaint-0.0.6
./configure
make all
```

While the programs are compiling, you should take a moment to add a NetSaint login, the user ID NetSaint will run as, if you have not already done so. Then, when the compile is done, you may finish the install. Here are the remaining steps:

```
adduser netstaint
make install
make install-config
make install-daemoninit
```

Quite a mouthful, *non*? I think we need a little something to wash that down with. François, refill our guests' glasses. *Merci, mon ami*.

Before you can use the package through the web interface, you will need to add a couple of lines to your httpd.conf file and restart Apache. The first of those

lines is as follows. Yes, I know there are two, but the one you need to add is the first. I'll explain in a moment:

```
ScriptAlias /cgi-bin/netsaint/ "/usr/local/netsaint/sbin/"
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
```

I showed you these two lines to illustrate an important point: make sure the NetSaint ScriptAlias appears before the standard cgi-bin ScriptAlias. This is very important. After these, you will now add an Alias definition for the main NetSaint HTML files:

```
Alias /netsaint/ /usr/local/netsaint/share/
```

Look in /usr/local/netsaint/etc to find a few scripts that will need modification for your site. Specifically, you will want to look at the hosts.cfg and the nscgi.cfg files, which were created when you did your **make install-config** earlier. Conceivably, you could have created them by hand, but that may have required more wine than we have in the cellar, *non*?

In order to access NetSaint functions with your browser, you must first authenticate with a user name and password. Do this by creating a .htaccess file in NetSaint's program directory, /usr/local/netsaint/sbin. I'll show you the file in a moment, but before we can use .htaccess, we need to add this section to the httpd.conf file, after the ScriptAlias information listed above:

```
<Directory /usr/local/netsaint/sbin>
    AllowOverride AuthConfig order allow,deny allow from all
    Options ExecCGI
</Directory>
```

Now, save the file and create the .htaccess file. This is what it should look like:

```
AuthName        "NetSaint :  Restricted access"
AuthType        Basic
AuthUserFile    /usr/local/restricted/.netsaint
<Limit GET POST>
   require valid-user
</Limit>
```

The AuthUserFile parameter points to the location of the actual password file. Notice that I have mine sitting outside of my web server's document root. In that way, the file cannot be downloaded with a simple HTTP request. Passwords are added to this file with the htpasswd command, like this:

```
htpasswd -c passwordfile username
```

Incidentally, if you built your Apache server from source, the program will likely be in the /usr/local/apache/bin directory. A note of caution here—the -c flag creates a new password file if none existed before. You do not want to use this unless you want to flush the whole file and start over. Another flag you may

want to use (and probably should) is the -m flag, which forces MD5 encryption on the passwords:

```
htpasswd -m /usr/local/restricted/.netsaint netsaintadmin
```

It's time to restart Apache. If your Apache installation is a source install, you likely will execute **apachectl stop** followed by **apachectl start**. That command is usually in the /usr/local/apache/bin directory. On something like a Red Hat system, you usually stop and start the httpd dæmon with a call to /etc/rc.d/init.d/httpd, specifying stop or start on the command line.

Now comes the fun part. Go back into the /usr/local/netsaint/etc/hosts.cfg file and start defining the hosts you want monitored. As I mentioned earlier, this is definitely one file where it makes sense to take what is there and slowly, host by host, modify it to suit your needs. Finally, you will want to grant yourself some permissions. By default, NetSaint is somewhat restrictive in what you can and cannot do. You grant yourself (possibly others) permissions by uncommenting the appropriate lines in the nscgi.cfg file. These lines all start with “authorized_for” and then specify a list of user names and what they can and can't do. That way, you can create multiple users and share the administration load. Here's one of these lines:

```
authorized_for_configuration_information=netsaintadmin
```

Mon Dieu! All this work, it makes for a big appetite, *non*? *Oui, mes amis*. It is time to start the NetSaint dæmon:

```
/etc/rc.d/init.d/netsaint start
```

If all has gone smoothly, we should be able to connect to NetSaint through our browsers. No special port numbers here. Assuming a system called my_system, you would use the URL http://my_system/netsaint/. For a peek at our NetSaint in action, I invite you to sample Figure 2.

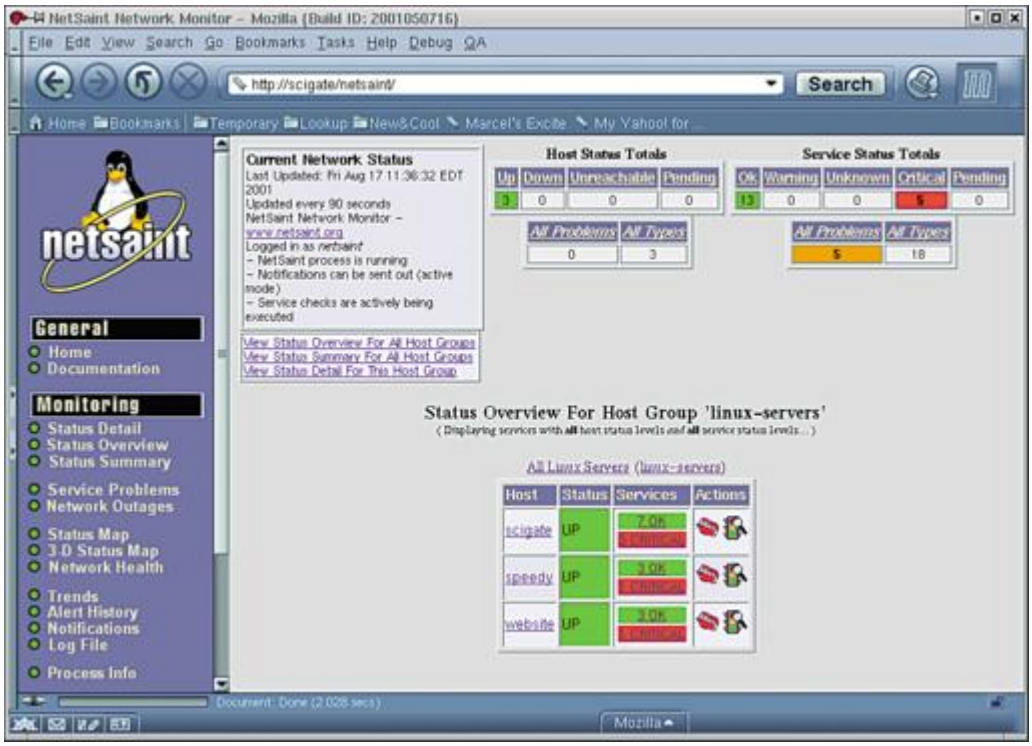


Figure 2. NetSaint up and Running

The NetSaint interface is rich and provides plenty of opportunity for exploration. You can customize things from here as well, and there is full, on-line documentation should you run into problems. Exploring every aspect of the program would take far more time than we have, but with your own NetSaint watching over your network, you should have plenty of time to explore later.

Ah, *mes amis*. The clock, she is telling us once again that we must finish our wine and head home. Since there is but a little left in the bottles, I shall have my faithful waiter refill your glasses a final time before you go. Until next time; *au revoir, mes amis*. Your table will be waiting.

A votre santé! *Bon appétit*!

Resources

email: mggagne@salmar.com

Marcel Gagné (mggagne@salmar.com) is president of Salmar Consulting Inc., a systems integration and network consulting firm, and the author of Linux System Administration: A User's Guide, published by Addison-Wesley.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Detecting Suspect Traffic

Michael Rash

Issue #91, November 2001

Use psad with ipchains/iptables rulesets to find TCP and UDPadvanced port scans and other such network baddies.

With the release of the much-awaited Linux 2.4.0 kernel early this year, GNU/Linux received a major boost toward entering the realm of enterprise-class operating systems. Several areas of the kernel have been improved since the 2.2.x series, not the least of which is the firewall code. Netfilter in the 2.4.x kernel series replaces the old ipchains firewall code in the 2.2.x series and incorporates many features that are required in an enterprise-class firewall, such as statefullness, DoS protection/rate limiting, network address translation (NAT), MAC address filtering and, last but not least, TCP packet filtering and logging based on arbitrary combinations of TCP flags. By contrast, ipchains (in addition to not being stateful, etc.) have the lamentable limitation of being able to distinguish between only two types of TCP packets: those that set and those that do not set the SYN flag. Netfilter's ability to distinguish arbitrary TCP flag combinations allows advanced port scans, such as those that are easily leveraged against a machine via nmap, to be detected. First we need a bit of background on nmap so that we can illustrate both how advanced port scans work and how to detect them.

nmap

nmap is the world's best-known port scanner and incorporates many advanced port scanning techniques that can be used to detect open ports on a target machine and/or network. In addition, nmap incorporates operating system fingerprinting and TCP sequence number prediction (among other things) to give better reconnaissance information than can be gleaned by determining which ports are open on a target machine. Three particularly interesting types of TCP scanning modes that nmap supports are FIN, NULL and XMAS scans. In normal TCP traffic, a FIN packet (or packet that has the FIN flag set) can be sent by either end of a TCP connection indicating that the session is finished or,

more precisely, that there is no more data to be sent. A FIN scan works on the principle that an orphaned FIN packet (i.e., one that is not already part of an existing TCP session) sent to an open TCP port should not be acknowledged. If such a packet is sent to a closed port, however, a normal unfiltered TCP stack is supposed to respond with an RST packet. Hence, to scan a host with FIN packets, nmap simply sends a single FIN packet to each of the target ports and waits for RST packets to trickle back. Ports that do not respond with an RST are open (or filtered by a firewall). Both the NULL and XMAS scans function similarly, but instead of setting only the FIN flag, the XMAS scan also sets the URG and PSH flags, whereas the NULL scan utilizes TCP packets that have no flag set whatsoever.

Setting up Netfilter

The basic principle in setting up a secure firewall is to have a default deny stance in which all traffic not explicitly allowed by the firewall is denied or rejected. In addition, the firewall should be configured to log any unauthorized packets to a log file for analysis by the system administrator or by automated logfile watchers such as psad.

For more information on getting iptables to run on your system, I highly recommend Rusty Russell's “Netfilter HOWTO”, available at netfilter.samba.org.

```
# iptables -L
Chain INPUT (policy ACCEPT)
target  prot source   destination
ACCEPT  all  anywhere anywhere state RELATED,ESTABLISHED
ACCEPT  TCP  anywhere anywhere TCP dpt:ssh flags:FIN, SYN,RST,PSH,ACK,URG/SYN
ACCEPT  TCP  anywhere anywhere TCP dpt:www flags:FIN, SYN,RST,PSH,ACK,URG/SYN
LOG     TCP  anywhere anywhere LOG level warning prefix `DENY '
DROP    TCP  anywhere anywhere
```

Stopping Port Scans

Does such an iptables policy stops port scans? First, any TCP packet (regardless of TCP flags) sent to a destination port other than 80 or 22 will be logged by rule number four and then dropped by rule number five. This takes care of any noisy scans of any type against any of the 65535 TCP ports other than 80 and 20. What about scans that test whether 80 and 22 are open? The answer is that it depends upon the type of scan. Rules three and four accept packets that have the SYN flag set and all other flags cleared, so any scan that makes use of just SYN packets will succeed. Also, any scans that make use of normal TCP connect() system calls, as a normal web browser or SSH client would use, will succeed since rule number one allows three-way TCP handshakes to complete.

nmap supports both of these scanning techniques with the -sS (half-open or SYN scan) and -sT (TCP connect() scan) command-line switches. Any other scanning technique that is not based on legitimate TCP traffic to these ports will

be blocked and logged. Such scans include the FIN, XMAS and NULL scans mentioned in the nmap section, as well as SYN/FIN and ACK scans.

Now that we are reasonably confident that our iptables firewall will block port scans, we should not become complacent with our success. It is not enough just to block scans; we should also do our level best to detect them, as a port scan could be a precursor to a more advanced attack.

Introducing psad

The Port Scan Attack Detector (psad) is a program written entirely in Perl and designed to work with restrictive ipchains/iptables rulesets in order to detect both TCP and UDP port scans. It features a set of highly configurable danger thresholds (with sensible defaults provided), e-mail alerting, optional automatic blocking of offending IP addresses via dynamic reconfiguration of ipchains/ iptables firewall rulesets, and verbose alert messages that include the source, destination, scanned port range, begin and end times, dns and whois lookups. For iptables firewalls, psad makes use of the extended logging capability to detect highly suspect scans, such as FIN, XMAS and NULL, that are easily leveraged against a target machine via nmap. In addition, psad incorporates many of the TCP and UDP signatures included with the Snort intrusion detection system to detect scans and/or traffic to various backdoor programs (e.g., EvilFTP, GirlFriend, SubSeven) and DDoS tools (mstream, shaft). **psad** is free software released under the GNU Public License and is available from www.cipherdyne.com.

The basic task of psad is to make use of firewall log messages generated by either ipchains or iptables to detect suspect network traffic. To accomplish this task, psad needs a way of efficiently getting the data it needs from the log messages the firewall writes to syslog. Hence, upon installation psad creates a named pipe called psadfifo in the /var/log/ directory and reconfigures syslogd to write all kern.info messages to the pipe. In syslog parlance both ipchains and iptables log messages are reported via the kern facility at a log level of info. The bulk of the work done by psad is accomplished by two separate dæmons: kmsgsd and psad.

kmsgsd

The kmsgsd dæmon is relatively simple and its sole responsibilities are to open the psadfifo pipe, read any kern.info messages that indicate ipchains/iptables has denied or rejected a packet and write all such messages to the psad data file /var/log/psad/fwdata. Since the regex included within kmsgsd looks only for packets that have been denied or rejected, the fwdata file provides psad with a distilled data stream that contains information that may be significant exclusively from a security perspective. However, this information is only as

complete and verbose as the firewall is able to generate, hence the need for the most restrictive possible firewall ruleset.

iptables does not support a logging option for any rule that drops or rejects packets. An easy solution to this, though, is to precede any drop rule with a logging rule that uses the --log-prefix option. See rule number four of simplefirewall.sh. The kmsgsd code snippet in Listing 1 illustrates reading firewall messages from the psadfifo named pipe and using the regex to match any dropped packet messages generated by either ipchains or iptables.

Listing 1. Reading Firewall Messages with kmsgsd

psad

Once the fwdata file is populated with packets that have been denied, it is the responsibility of the psad dæmon to analyze and make judgments about whether or not the packets constitute a port scan or other suspect network traffic. **psad** accomplishes this by periodically checking fwdata for new lines indicating that packets have been denied recently by the firewall. Port scans are assigned a danger level from one to five based on how many packets are denied within a fixed period of time. However, a scan may also be assigned a danger level depending on whether it matches any of the set of nasty signatures contained within the psad_signatures file. Examples of such signatures include “NMAP Fingerprint attempt” (for any packet that sets the URG, PSH, SYN and FIN flags) and “DDoS - mstream client to handler” (for a SYN packet destined for port 15104).

After a port scan reaches a predetermined danger threshold, an e-mail is sent that contains several pieces of information: the source IP from which the scan originated, the destination IP, the range of newly scanned ports (TCP or UDP) within the last scan interval, the complete range of scanned ports since the beginning of the scan (TCP and UDP), the start and end times, the danger level assigned by psad, reverse dns information, the TCP flags that were used in the scan (along with the corresponding nmap command-line options that would generate such as scan) and whois information.

Instead of appealing to the whois client installed by default on any Linux distribution, psad makes use of an excellent whois client written by Marco d'Itri. This client has the advantage of accurately querying the correct whois database for almost any scanning source IP address. **psad** also features the ability to reconfigure an ipchains/iptables ruleset to block any IP address that has reached a sufficiently high danger threshold. This feature is disabled by default, since many administrators do not want volunteer administrators across the Net having the ability to force the firewall to block access to any web site they choose by spoofing nasty packets from the web site's IP address.

Some administrators, however, like the ability to set a high threshold for a port scan or other nasty traffic and have psad block it automatically. Note that any traffic analyzed by psad has already been blocked by the firewall. But if a scan reaches a sufficiently high threshold, the auto-blocking feature will block all traffic from the offending source IP, since the administrator would probably not want such an IP sending any packets to the local network, legitimate or not.

psad also includes a signal handler such that if a USR1 signal is sent to the psad process, psad calls Data::Dumper to dump the contents of the %Scan hash to the /var/log/psad/scan_hash.$$ file, where the $$ represents the pid of the psad process. The %Scan hash is the main psad data structure and contains all scan information, and hence dumping its contents is useful both as a way to capture a record of scan data and as a debugging tool for extending psad's feature set. **psad** is a work-in-progress with much on the to-do list, but it is being developed actively and also has a relatively short version-release cycle.

ipchains vs. iptables Logs

To illustrate the differences in the output of ipchains and iptables firewalls, we first compare log entries generated by an nmap XMAS scan.

ipchains

The ipchains messages in Listing 2 were generated by an nmap XMAS scan of TCP ports 79 through 81. Recall that an XMAS scan sets the FIN, URG and PSH flags. First the nmap command and output is listed followed by the corresponding ipchains output. Note that ipchains makes no mention of which TCP flags were set.

Listing 2. nmap Command and Output with ipchains Output

iptables

Listing 3. iptables Messages from TCP Session with Web Server

Now we perform the same nmap scan (the nmap command line and output is identical to the above ipchains example, so it is not repeated) and display the corresponding iptables output (see Listing 3). This time we can plainly see the FIN, URG and PSH flags set in the packets used in the scan.

Resources

email: mbr@cipherdyne.com

Michael Rash works as a senior security engineer for an ASP in Annapolis, Maryland. He holds a Master's in Applied Mathematics from the University of Maryland and has been tinkering with Linux since 1998. He can be reached at mbr@cipherdyne.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Implementing a Research Knowledge Base

Michael Yuan

Issue #91, November 2001

Keeping up with large volumes of research requires a systemboth flexible and intuitive.

Research has always been an integral part of education, especially higher education. Research generates new knowledge and offers training in creative and independent thinking. The role of research in higher education is shown by the fact that almost all the famous universities in the world are also known as research universities.

Modern research is highly specialized and highly collaborative. Finding papers with more than 100 references is common these days. Moreover, the rise of the Internet has made it possible to publish results quickly, and the barrier to accessing scientific information has been lowered drastically. For instance, in the astrophysics field alone, its internet preprint service publishes more than 15 new research papers every day, all of which are freely accessible to the public. While this provides enormous power for researchers to conduct advanced research, it also creates a huge burden to keep up-to-date with the newest results.

These conditions give rise to two important questions in modern research: how do we organize knowledge to keep in step with recent developments and be able to retrieve the information when we need it? Second, how do we organize and keep record of research projects involving several researchers? In the research world, new knowledge is presented in the form of publications. I use the term “references” to refer to pieces of knowledge, including unpublished or privately communicated tips and results. In this article, I discuss research knowledge-base systems in the form of reference management systems.

Why We Need New Software

There have been commercial attempts to address the questions above. Applications like EndNote, Pro-Cite and PAPYRUS offer reference management capabilities and even some web capabilities. However, I found them hard to use and customize in my own research.

Research covers every field of human knowledge, and more importantly, research is intended to explore the unexpected. Every group might have different requirements for organizing and displaying their references. Most of the proprietary reference management systems have targeted specific research fields (usually medical research). Being closed-source software, it is impossible for the user to change and improve the software to adapt to specific needs.

The best solution to this problem is to design an open-source, web-based, multi-user knowledge-base system. It would run on an internet-connected server and be accessible from any standard web browser, from all platforms. Users could post/organize references and hold discussions in the comment section. All the knowledge and discussion interactions could be archived centrally from one secured server. This web-based approach would use web browsers as the user interface (UI) and would enable anyone to change the UI by simply changing the HTML-like source code. My answer to the need for such software is the OpenReference reference management system.

Goals and Required Features

One goal for writing this software was to categorize knowledge for easy future retrieval by multiple users. A simple keyword search is not enough, since searching cannot guarantee finding all the related references. It is much easier to browse through the category tree if you have a specific subject in mind. So, advanced categories and user management are two core features I decided to implement.

Big categories, each containing hundreds of references, are no better than no category at all. For the categorization to be useful, the leaf categories on the tree should contain less than one page or 20 references each. That demands a very fine-tuned category structure. Finer categories are needed in areas of more active research. It is impossible, however, to judge in advance how many levels of subcategories are needed in any field so as to make efficient categorization. The only way is to design a dynamic category structure that can be adjusted at runtime. If a lot of references show up in a particular category, the administrator can divide it into several subcategories, according to the nature of those references.

As I have stated, the leaf categories need to focus on narrow subjects to keep the number of references small. Today's research works have become more and more interdisciplinary, making it hard to categorize a reference into a narrow category. The solution to this problem is to allow a reference to associate with more than one category.

This system is designed not only as a personal reference organizer, but also as a group discussion server to exchange ideas in the comments section. A web-based collaboration system can keep track of information and make archives of idea exchanges possible.

Being a multi-user system, this software must establish some user access control. The administrator can set the policy to accept new users. Every user needs to log in with a legitimate username/password combination to post references and comments. Each user can edit/delete/recategorize his or her own postings. Only the administrator can touch the category structure.

Finally, in order to enable private conversation in the forum, I also allow a user to specify a list of other users who can see her or his postings. Those private conversations traditionally take place in e-mail communications, but this software encourages users to use the web system for better archiving of the research effort.

Database

The back end of the system is naturally a relational database management system (RDBMS). Most of the data we want to store/retrieve/search is textual, and RDBMS is perfect for this purpose. Any SQL-enabled database server will work, and there are many such servers to choose from. I chose the GPLed MySQL for its speed and reliability. If data integrity and transaction support is a must for you, you can choose the open-source PostgreSQL database server.

Middleware

The middleware is a collection of classes/methods that wrap the data query operations. They are designed to shield front-end developers from the technical details of database connections and SQL language.

I chose to use Java to develop the middleware. One big advantage of using Java is that it is a full-blown, object-oriented language, making it much easier to implement complex logic/structure designs required by large projects. Using Java also allows us to take advantage of a large number of utility classes already existing as Java libraries or beans. The ones I used for this project include JDBC driver, database connection pool, session management and text processing.

The standard way to build database middleware in J2EE is to use entity EJBs (Enterprise JavaBeans). However, this approach requires running an EJB container, which can be expensive. In fact, few low-cost JSP hosting services provide EJB containers. For OpenReference's relatively simple database structure, I decided to use a simpler approach: static methods in helper classes to provide database access. Each row in a table is represented by a HashTable.

The middleware uses JDBC to pass information between the Java application and the SQL database. There are JDBC drivers for all the major RDBMSes. For MySQL, I used the mm.mysql driver. I constructed one class for each database table. The class knows the fields in that table and knows which fields are searchable. Each class implements a set of basic data query functions (e.g., getAllRows, AddRow, updateRows, etc.) and a search function that searches all the searchable fields and returns all the matched rows. Each class also has its own query functions to do specific or cross-table queries. For example, in the ReferenceTable.java class, there is a function getReferencesByUserName. This will take the user name as input and find the corresponding user ID in the User table, and then return the rows with matching user ID from the Reference table. As an example, see Listing 1 [available at ftp.linuxjournal.com/pub/lj/listings/issue91/4769.tgz] for the complete API for the Category class.

Front End

I chose JavaServer Pages (JSPs) for the front end. JSPs have the power of the full Java language plus the benefit of separating the web presentation from the application functions. JSPs support all the HTML tag syntax for formatting display, and one can add whole Java programs dealing with beans and other functions in HTML comments. One can even design custom tags to encapsulate the back-end operations (e.g., database queries). It is easy to train an HTML programmer/presentation expert to work on the web pages using their favorite HTML editor, without caring about how the database queries are executed. In the meantime, a back-end programmer can work on the data query part without caring about how the data will be displayed. More information about JSPs can be found in Reuven Lerner's At the Forge column in the May through July 2001 issues of *Linux Journal*.

Any J2EE-compatible Java server is capable of running JSPs. My favorites are the Tomcat engine from the Apache Foundation and the Resin engine from Caucho Technology. Either one of them can run as an extension module of the Apache web server to take advantage of many other useful features of Apache. Tomcat is released under GPL. Resin is closed-source software but is free for noncommercial use. Resin runs considerably faster than Tomcat and offers some useful features, such as a built-in JDBC driver and driver pool and multiple JVM for fallback.

Dynamic Category Tree

Originally I planned to use an XML file to represent the category structure because XML documents are naturally organized in a tree structure (DOM model), and they are easy for human reading and editing. There are many good XML-DOM tools in Java to manipulate trees.

However, we need a large and dynamic category structure for accurate classification and browsing, as it is important to be able to search the categories. One drawback of using XML is its difficulty to be searched together with other content stored in RDBMS. Also, to store a big parsed DOM object in memory constantly is inefficient and difficult to synchronize among several JVMs. To store it on disk and parse it when needed introduces too much processing overhead. So, I decided to use database tables for the categories.

The whole category is stored in a database table. Each record represents a category and has its unique category ID. It also has the parent category ID so that the records are linked together into a tree. References are linked to the categories by a separate category ID vs. reference ID table.

The Category class in middleware contains all methods to operate the tree. In order to maintain the links and structure integrity of the category table, it is important that we manipulate the category table only through the Category object public methods. Some important methods include:

- Insert new subcategories in or between any level(s) in the tree. A special note about adding a new subcategory under a leaf category: since all references must be associated with leaves only, references that were associated with the old leaf become associated with the new subcategory now.
- Change category description/keywords/properties.
- Delete a category from any level in the tree to make the target category's children be children of its parent and then delete the target itself from the table.
- Return a list of children (or parent) for a given category.
- Search keywords from category descriptions.

An illustrative listing of Category-class source code is presented in Listing 1. I found those methods sufficient for my use. You might want to do more complicated operations, such as moving a subtree to another branch, etc. The strength of open-source software is that anyone can add functions to the code without rewriting the basic part.

User Authentication and Sessions

The user login and sessions are managed through the session API from J2EE. Usernames and passwords are authorized from a database table, then the servlet engine establishes a session for this user. The servlet engine tracks the sessions through session objects.

The session object needs to know about the user who owns it. That includes the user name, profile, preferences and current browsing status. Storing such information in the session object improves performance a lot. Otherwise, for example, the program has to look up the database for display preferences every time prior to displaying a page for a user.

Instead, I could store all the information inside the session object itself. But for better organization, I used several JavaBeans associated with session objects to store additional user information. In JSP specifications, a JavaBean can have a scope of a session or a page. Sessions with JavaBeans greatly simplify the development work.

Database Connections

Database connections are expensive to establish. To open a new connection every time we need to query can slow down the computer drastically and use up the system resources quickly.

There are several ways we can reduce the need for new connections. One way is to assign one connection for each session. That connection can be stored in a bean with session scope, and all the queries from that session go through it. However, if many sessions are active at the same time, available connections run out fast. This solution does not scale well.

Another choice is to assign one connection per page. However, I do not like this idea; the connection object has to be passed to the middleware by the JSP coder. This is not intuitive to a nontechnical JSP coder and defies the goal of separating presentation from application logic. It is possible to design a set of custom JSP tags to encapsulate the database connections so that JSP coders do not see them, but it requires extra designing work and that the JSP coder learn a nonstandard language. Using custom tags certainly increases productivity in the long run, but it also makes short and simple changes more difficult with a steeper learning curve for the system. I am still seeking the best solution. For now, I decided to hide the connection handling completely inside the middleware.

I made a new database connection from every database query function in the middleware. Each query function completes multiple-related queries to make efficient use of connections. Each JSP page calls only one or two such functions.

To reduce the overhead caused by opening new connections, I used "connection pool" utilities. These utilities are classes that maintain a pool of open connections in memory. When the user requests a new connection, it simply fetches one from its pool rather than making a new one. When the user closes the connection, it returns to the pool. There are several such utilities, e.g., PoolMan. Their usage is very straightforward, and you only need minimal changes to convert your code to take advantage of those utilities.

Text Processing

We have to allow users to mix some HTML tags in the text so that they can format the memo and comments properly. However, displaying user HTML has a big security risk: unauthorized user HTML tags can corrupt/change the display of page navigation and other people's comments. They can then be used to load unauthorized images/applets or even redirect the user to another site using JavaScript.

So, I decided to allow only four HTML tags: <p>, <b>, <i> and <a>. That ensures the user cannot change the format of any content other than her or his own. The unauthorized HTML tags are filtered out before submitting text into the database. The allowed HTML tags are configurable by the site administrator at compile time.

It is also important to note that some users might want to input XML sample code or mathematical formulas containing "<" in memos or comments. It would be inappropriate to treat all "<" as HTML tag tokens. So, I provide the plain text mode under which all the "<" symbols are converted to "<" before being submitted into the database.

A powerful way to process text is using regular expressions. There have been several good Java regular expression engines available. I choose gnu.regexp because it is a free software implementation of almost all Perl5 regular expression features through a simple and intuitive API. The code for filtering HTML tags and composing the SQL query string using a regular expression engine is listed in Listing 2 [available at ftp.linuxjournal.com/pub/lj/listings/issue91/4769.tgz].

Resources

email: juntao@astro.as.utexas.edu

Michael Yuan is a PhD candidate in Astrophysics at University of Texas at Austin. He studies remote quasars (20+ billion light years away) to understand the history and evolution of our universe. When he is not observing quasars, he enjoys developing useful software using earthly languages such as Java and Perl.

Archive Index Issue Table of Contents

Advanced search

XFree86 4.1.0 and ATI RADEON

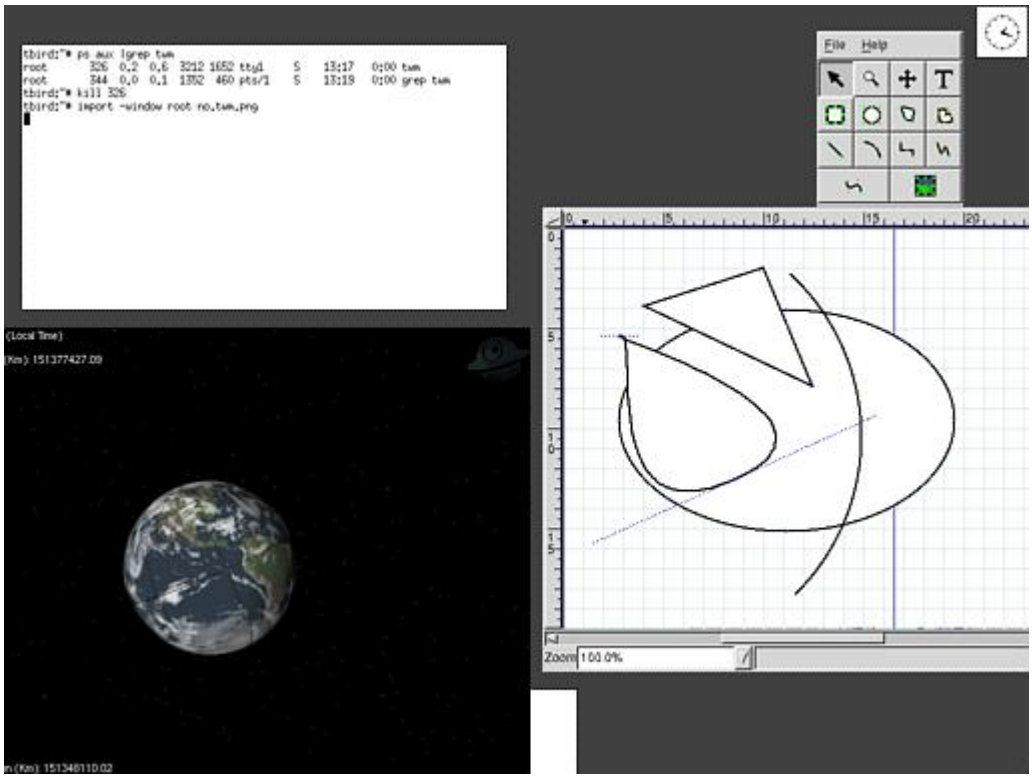
Robin Rowe

Issue #91, November 2001

A brief history of the X Window System and the current stateof X-related graphics drivers.

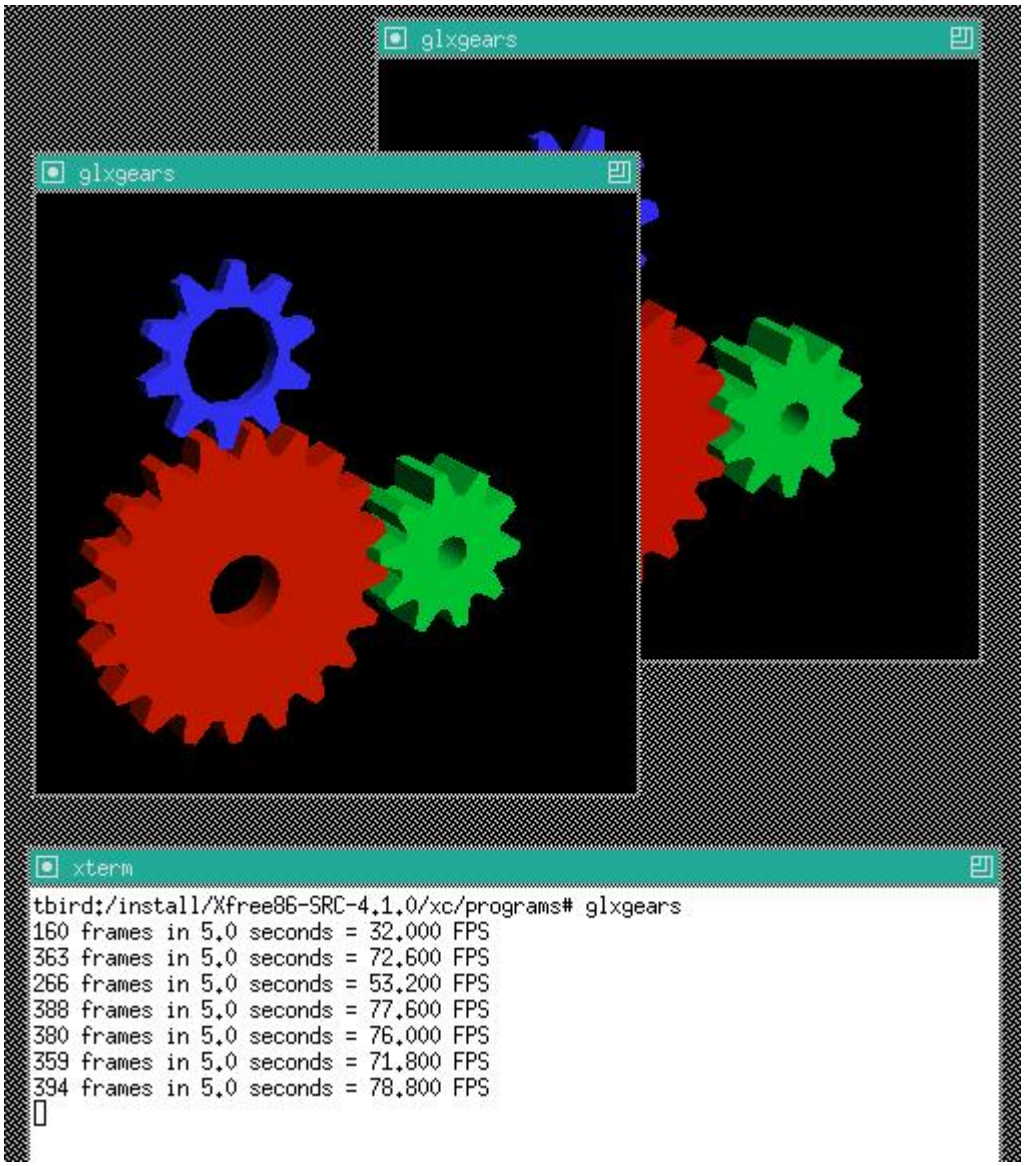
Linux graphic performance is significantly improved by the latest version of XFree86, making Linux viable as a workstation operating system. Major motion picture studios, including DreamWorks and Pixar, are adopting Linux as their standard movie production workstation.

XFree86 is a popular implementation of the X Window System, a technology developed in 1984 at MIT as part of Project Athena. MIT's goal was to build their existing assortment of incompatible workstations from various manufacturers into a network of graphical workstations. At first X was used primarily at MIT and DEC where much of the development also occurred, but in 1986 it was commercially released. Eventually MIT handed X off to the X Consortium, which later folded into Open Group. In 1999, Open Group spun off X.org. X.org estimates that more than 30 million people are using X, and that number is growing rapidly thanks to Linux and XFree86.



Deliberately killing the twm reveals what happens if you lose your window manager. Our Dia, OpenUniverse and xterm windows still display, but there is no means to move them.

In 1992, after X had stagnated for a decade, the project that became XFree86 was founded with the goal of offering a free implementation of X. The significant new features of X (font anti-aliasing, compositing and DRI) are due to XFree86. A volunteer organization, anyone may join XFree86 and gain direct access to the XFree86 code and docs in CVS. Originally focused on Intel x86, XFree86 today has the goal of being the best windowing system on every platform available. It runs with Linux, the BSDs, Mac OS X, Solaris and OS/2. CPUs supported include x86, Alpha, PowerPC and SPARC, with MIPS in development.



glxgears

In March 2000, a redesigned XFree86 was released. Version 4.0 added much better graphics acceleration plus features such as X-Video and multihead support with Xinerama. X-Video provides support for color space alternatives to RGB, such as YUV used in television and movies. Animators, in particular, often need to work with more than one monitor at a time. Ordinary multihead configuration supports placing windows on multiple monitors. Xinerama enables one window to span multiple physical screens simultaneously.

XFree86 3-D graphics performance is significantly improved by the addition of direct rendering infrastructure (DRI) and GLX. OpenGL is a platform-independent, scene-description language, well suited for animation and closely tied to hardware. GLX is the window-event code that connects the driver-level OpenGL graphics language with X. GLX and OpenGL were both developed by SGI. DRI is a pipeline that bypasses inefficiencies in the X server to bind the kernel, X server, OpenGL and graphics drivers together directly. All modern

high-performance graphics boards now offer accelerated drivers for Linux, with the notable exception of 3D Labs, which has a driver in development.

A particularly handy feature when configuring XFree86, VESA display data channel (DDC) enables XFree86 to query modern monitors for their supported screen resolutions and refresh rates. XFree86 4.x integrates the read-edid utility to determine the correct monitor mode-line settings automatically.

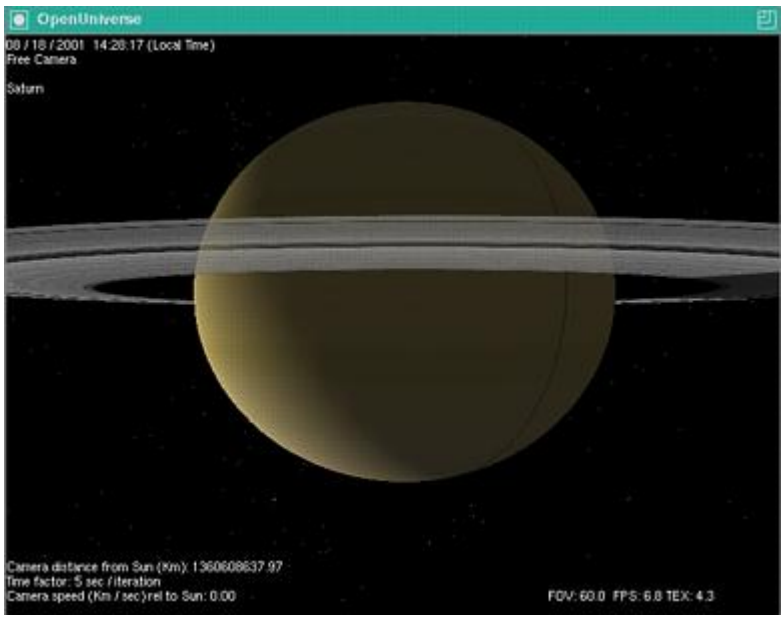
In June 2001, XFree86 4.1.0 was released. Besides bug fixes, this version has many more supported drivers; of particular interest to us is the ATI RADEON. All the RADEON cards have excellent graphics performance, but the All-In-Wonder version of RADEON has the added feature of a built-in TV tuner. ATI, who had a bad reputation with open-source developers, went from being the worst to one of the best. ATI sponsored the development of the RADEON open-source driver.

Daryll Straus built the first accelerated Linux 3-D driver (for 3dfx hardware) in his spare time while working at Digital Domain on the renderfarm for *Titanic*. That was the first major film to have its effects rendered primarily using Linux. (See “Linux Helps Bring *Titanic* to Life” by Daryll Strauss and Wook in the February 1998 issue of *Linux Journal*.) From there, Strauss went on to build Linux graphics drivers as the lead for multimedia at Precision Insight, now a part of VA Linux. Precision Insight was funded by ATI, Intel, Matrox and 3dfx to build DRI-compatible Linux drivers. Strauss says:

> I think that open source is the way to go. In 18 months we built ten drivers: 3dfx Voodoo 3, 4 and 5; ATI Rage 128, 128 Pro and RADEON; Matrox G400; Intel i810 and i815; and another not yet announced. Creating a common code base really helped. In looking at the source and documentation of proprietary drivers under NDA, we discovered that the vendors' code and hardware share many features.

His group did a limited implementation for RADEON, without acceleration for 3-D transformation and lighting. “The thing that's annoying about the RADEON is we created a rasterization driver, effectively turning RADEON into a fast Rage 128”, says Strauss. “As soon as you run Maya you notice you don't have the 3-D performance.” Maya, reviewed here last month, is a popular 3-D modeling application for motion picture film animators.

OpenUniverse lets you voyage through a simulated solar system, another test of graphics performance.



More graphics with Open Universe

XFree86 may be installed three ways: as a package, as a binary or from source. The packaged version wasn't available yet, so we couldn't follow that route. We tried to install from binary, but had problems. The Xinstall.sh install script didn't like the space in the name of the directory it was installed in, but more than that, we just couldn't get it to work right. Building from source seemed our only option, an interesting though challenging process. Version 4.1.0 relies upon the 2.4 kernel. We upgraded not only to kernel 2.4.7 but to Debian Sid. If you would

like to read the details of that process, including several unexpected problems we overcame along the way, read the on-line version of this article at www.linuxjournal.com.

Installing XFree86 4.1.0 won't seem like such a challenge when it becomes widely packaged, but building it from source taught us a lot about how XFree86 is put together. The sheer size of it and the complexity of the build process is more than one expects. Vladimir Dergachev, who regularly helps RADEON users on the XFree86 Xpert list, was indispensable in helping us get it to work. He's one of the volunteers actively working on the RADEON driver.

The new DRI architecture with accelerated drivers (both open- and closed-source versions) make Linux a much more powerful platform for graphics. ATI, HP, NVIDIA, 3D Labs and others are actively working on closed-source drivers. What's not happening is further development of open-source drivers, such as the RADEON driver described here. Precision Insight has gone on to other projects, and no one is currently funding enhancements to open-source drivers or even to XFree86 itself.

The learning curve for volunteers to write DRI-compatible drivers is steep, especially because of the lack of documentation. With version 4.1.0, the changes have settled down in XFree86. It is time to write good documentation that will enable future programmers to build and enhance accelerated drivers. If anyone would like to help, please drop me a line.

Resources

email: Robin.Rowe@MovieEditor.com

Robin Rowe is a partner in MovieEditor.com, a technology company that creates internet and broadcast video applications. He has written for *Dr. Dobb's Journal*, the *C++ Report*, the *C/C++ Users Journal* and *Data Based Advisor*. He may be reached via e-mail at robin.rowe@movieeditor.com.

Archive Index Issue Table of Contents

Advanced search

Applications for Your Enterprise

David A. Bandel

Issue #91, November 2001

If you think Linux isn't ready for the enterprise, thinkagain. Linux in the enterprise is alive and growing.

I'm continually surprised at the folks who (with an air of authority) declare Linux is not ready for the enterprise. These are people who should know better. I told you several months ago that Linux was, in fact, ready for the desktop. True, there are some areas where Linux is still weak. But I think some folks are judging Linux from a whole different set of standards than they use for other operating systems. I submit that the Burlington Coat Factory, McDonald's and Largo City success stories prove that Linux is ready. And with over 400 Largo City desktops running Linux, with a savings of over $3,000,000 US, and the secretaries loving it, folks need to start examining why they're fighting it so fiercely. Linux in the enterprise is alive and well and growing, with many happy users. The absurd claim that when those who put it together are gone no one will understand, it is wishful thinking. I will bet most anyone reading this, given one or two hours (or less) going over the configuration files in /etc, would understand everything going on. So don't ask why, ask why not.

MySQL Navigator sql.kldp.org/mysql

If you find you need a powerful graphical tool to do serious MySQL work, then MySQL Navigator may be just what you need. This application rivals, and maybe even betters, PHPMySQL. And if you don't have access to Apache with PHP and MySQL support, then this is definitely what you need. Requires: libmysqlui (included), libqt2, libXext, libX11, libstdc++, libm, glibc, libmysqlclient, libSM, libICE, libpng, libz, libjpeg, libmng, licvrypt, libnsl.

Mindi www.microwerks.net/~hugo/mindi/index.html

A number of recovery CDs have been appearing, but this one is created from your running system. While Mindi is comparable to several other offerings, it is

quick, easy to use and worth a look. Its minimal requirements make it perfect for almost any system, including one that doesn't have its own CD burner—just transfer the ISO image to a system that does. Requires: sh.

Mondo www.microwerks.net/~hugo

If you need more than just a boot CD-ROM, Mondo extends Mindi and creates a backup of your entire system. It first creates a Mindi ISO, then calculates and creates a backup of your entire system that is easy to restore. Those of you comfortable with afio will probably like this backup system, which is another good tool to consider for your backups if you use a CD burner rather than tapes. Requires: Mindi, afio.

tcptraceroute michael.toren.net/code/tcptraceroute

Ever had a problem where you needed to trace all the way to a server but that server was behind a firewall that dropped standard traceroute packets? Well, if that server is running services accessible from the Internet, you can use tcptraceroute to go right through the firewall to the server. Just specify the port (by default, tcptraceroute uses port 80), and the firewall won't know the difference. Requires: libpcap, libnet, glibc.

iCE Breakers Log Monitor sourceforge.net/projects/iblm

I've tried a number of different tools to keep an eye on certain logs as events happen, but this one is by far the best one I've seen. Just tell the dotrc file which logs to watch for you (you'll obviously need permissions to read these files), and you can watch any one by selecting its tab. As messages come in, if they don't come in to the tabbed box on top, the lettering on the tab changes to bold to alert you of a new message. A great tool for troubleshooting. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.

WebHost Billing System www.tolchz.net

Granted, not many will have a need for this, but if you bill a lot of customers monthly, you might want to give this a try. It's particularly good if billings go out on a daily basis. Just select the customers to bill and off it goes, tracking who's been billed what and when. Requires: MySQL, PHP4 with MySQL, web server, web browser.

FAQ-O-Matic faqomatic.sourceforge.net

I don't know if you've ever tried to manage a FAQ, but it can be a challenge. Well, the FAQ-O-Matic won't do it for you, but it does make it easier to maintain. It's all done from a web interface, so it's relatively painless and easy. Installation

is a breeze. In true Perl fashion, this little utility runs you through the entire process, checking off boxes for you as you go. Requires: Perl.

toolbox movingparts.windsofstorm.net/code.shtml#toolbox

I realize that only a very few of you out there run the Blackbox window manager. But for those of you that do, toolbox can help you customize your Blackbox setup easily. While I'm sure Blackbox users are more comfortable with ASCII configuration files than the average KDE user, the styles are more easily modified via a graphical interface. At least, that's how I see it. Requires: libqt (v2), libXext, libX11, libstdc++, libm, libSM, libICE, libpng, libz, libjpeg, libmng, glibc.

Until next month.

email: david@pananix.com

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

Embedded Linux: a Timely New Book

Rick Lehrbaum

Issue #91, November 2001

A review of the first embedded Linux book to hit the shelves—a must-read for anyone interested in this subject.

I first met John Lombardo at the Embedded Internet Conference in August 2000. John told me he was working on a book about embedded Linux, which would be published within a year.

In contrast to the inevitable lateness of so many software releases, Lombardo's *Embedded Linux* (New Riders, ISBN: 073570998X) hit the shelves of bookstores right on schedule. As far as I know, it's the first and, at the time of this writing, the only book published on this timely subject, thrusting it very much into the spotlight and making it a must-read for all developers (and many others) interested in exploring the subject of embedding Linux.

Embedded Linux covers a lot of territory. There are four main sections: software, hardware, application development and supplementary references. These sections are further organized into ten chapters and four appendices, including chapters on booting embedded Linux devices, hardware considerations and testing and debugging.

The complete, detailed table of contents is available on-line at www.newriders.com/tocs/073570998X.pdf. A sample chapter was published by *Embedded Linux Journal* in the May/June 2001 issue.

This 192-page book can be read comfortably in less than a day, providing you don't launch into the step-by-step Embedded Linux Workshop exercise (more on that later).

An exhaustive work on the subject of using Linux in embedded systems and smart devices certainly could occupy a lot more pages. Obviously, picking the right 192 pages' worth of information and presenting it in an interesting, useful

and well-written manner must have presented a real challenge to the book's author—one that he handled quite skillfully.

Another challenge, when you're putting together the first book covering a hot new disruptive technology like embedded Linux, is the race against time. First, there's the pressing need for a book on the subject. Although several embedded Linux workshops and training sessions are now available, only a few developers can actually manage to attend them—a technically oriented book on embedded Linux is in demand. Students, professionals and interested onlookers are hungry for something to educate themselves about embedded Linux. Another reason time is critical when producing a book like this is the subject matter itself undergoes extremely rapid change. Companies rise and fall. Distributions come and go. New tools and capabilities arrive on the scene almost daily. The time cycle in getting this type of book to market must be as short as possible.

The book starts off with an excellent introduction to the idea of using Linux as an embedded operating system, proceeds to outline the options available for embedding Linux and then launches into valuable discussion of how to design an embedded-based system on Linux, including software, hardware and system-level considerations. The book is chock-full of good, practical advice for embedded developers in general, and especially for embedded Linux developers.

After covering all the requisite software and hardware fundamentals, Lombardo then settles down to the real meat of his book: the Embedded Linux Workshop (ELW). The ELW is an open-source embedded Linux toolkit that Lombardo created as a means to simplify the process of building Linux-based embedded applications and to serve as a companion project for his book.

Over 35 pages (including all of Chapters Seven and Eight) are devoted to the ELW, including step-by-step instructions on how to download the ELW toolkit and use it to implement a simple but real embedded project—embedding Minicom. If you're interested in really learning how to embed Linux, you'll find the sections on the ELW to be extremely valuable as a way to understand fully the basics of embedding Linux. The entire ELW toolkit is open source and can be downloaded, configured and tested on any desktop PC. The ELW is structured in a manner that the entire build process and all of the associated software are exposed fully to study and scrutiny. What a perfect project for a book like this!

Lombardo is clearly an embedded system developer, not a marketing guy, and this book is about as far from marketing fluff as you're likely to get. There are even sections called “Disadvantages of Open Source” and “When is Linux

Inappropriate?" thrown in for balance. You certainly don't get the sense that you're reading the words of a religious zealot.

In the words of the book's back cover, "*Embedded Linux* is intended for designers of embedded systems and information appliances, as well as for general Linux programmers." This includes not only software developers, but system developers as well. Even hardware engineers can find much of value in this book, thanks to several sections devoted to system design issues and debug considerations. It's also full of useful information on the techniques and issues associated with developing and supporting embedded systems in general. In short, I think anyone wanting to become more familiar with the concepts of embedded system development can benefit from reading this book, although nonprogrammers will find themselves skipping past several large chunks of code-intensive material. (Being more of a system/hardware guy, I just close my eyes during the scary software parts.)

Overall, I found *Embedded Linux* to be easy and enjoyable reading (other than the aforementioned scary parts) and packed with useful information. It provides insight into embedded system development issues—good coverage of fundamentals of embedded system architecture design, including things like selecting hardware and software, system architecture trade-offs, make vs. buy issues, etc. It's also a good source of great tips/techniques/tricks on embedded software development—the author shares his considerable experience as an embedded system developer throughout the book, in the form of a steady stream of suggestions, tricks and caveats to help developers simplify and accelerate their embedded projects.

On the negative side, the book has a limited breadth of resources. Although it provides a good general introduction to embedded Linux, it seems a bit thin when it comes to lists of options and alternatives. Perhaps it would have been better to focus less on a single commercial embedded Linux toolkit (LynuxWorks BlueCat) and instead provide brief overviews of perhaps half a dozen of the major alternatives, along with their differentiating characteristics (a comparison table of distros would be really nice).

Potential for confusion exists between embedded Linux "distribution" and "toolkit"--in my opinion, the author blurs the distinction between embedded Linux distributions, such as Lineo Embedix, MontaVista Hard Hat and LynuxWorks BlueCat, and embedded Linux toolkits. He defines toolkits as "designed to simplify the job of building the binary that runs your device". Products such as Embedix, Hard Hat and BlueCat, which are called toolkits in the book, contain both a Linux OS and toolkits for building target OS binaries. The Embedded Linux Workshop is appropriately termed a toolkit because it doesn't include things like the Linux kernel, compiler, libraries, GNU utilities,

etc. For that reason, I would prefer to see the commercial products called embedded Linux distributions, since they include both toolkits and the GNU/Linux OS.

Also, ELW coverage needs restructuring—given the extraordinary utility of the ELW, I would suggest that it might make sense to break the chapters on the ELW out into a separate section of the book and expand that coverage a bit. I suspect the ELW could be the subject of its own 192-page book, especially once its SourceForge project takes off.

In summary, I give John Lombardo's *Embedded Linux* a “3.5 Tux” rating (on a scale of 4). It's first-to-market, well structured, has a lot of useful information and deserves high accolades for the excellent Embedded Linux Workshop and associated step-by-step examples.

Resources

email: rick@linuxdevices.com

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

Archive Index Issue Table of Contents

Advanced search

Original and Instant

Doc Searls

Issue #91, November 2001

Thoughts about Jabber's youthful and enthusiasticdevelopment team.

At the first break on the first day of Jabbercon, four members of the development team made their way over to say hi. The first to arrive was Eliot Landrum who told me how much he liked the new *Linux Journal* design changes, for which I thanked him. Then Julian Missig, Justin Mecham and Schuyler Heath followed. They looked like a high-school chess team, but by now I've learned never to let looks deceive me. These guys were short in the tooth but long on programming skills, and they helped make Jabber something worthy of its first conference.

I asked them how they described Jabber to people who never heard of it before. “Distributed open-source instant messaging”, Eliot said. The others kind of nodded along, but their expressions said “whatever”. Which is kind of true, too.

In fact, they were far more curious than opinionated. Eliot said something like, “We've been sending out these messages into space, and we want to see what bounces back from other civilizations.” Among the civilizations he's talking about are businesses excited about what he and his cohorts have created. On the attendee list I saw emissaries from Intel, Earthlink, Disney, PeoplePC, Swiscom and a bunch of less familiar companies, some of which were reportedly doing some very interesting stuff. One with a lot of buzz was Nuance, which lets you use your cell phone as an IM (instant-messaging) client in a real-time, speech-to-text-and-back-again conversation over a Jabber transport. Another was Oracom, which uses Jabber for worldwide network management in real time.

Both of these are examples are of what we might call “embedded IM” if that label didn't seem to exclude XML routing and presence management—to name

a couple other virtues that make Jabber easily embeddable and so far outside the AOL-defined meaning of IM that it begs a whole new label.

“Platform” isn't it. As with embedded Linux, Jabber is deeply structural, yet too dedicated to arcane purposes to justify the “platform” label, which now more legitimately belongs to the Net than to any operating system. Jabber's job is not to make its own presence known (or in dot-com parlance, to “brand” itself) but to add presence and data routing (by XML) to the growing roster of fundamental and ubiquitous internet services.

So I look at these guys and ask a rude question: “How old are you?” “Nineteen”, says one. “Seventeen”, says another. “Nineteen.” “Twenty-one.”

“How old is Jeremie?” I ask. Jeremie Miller is the Linus of Jabber, the programmer whose itch Jabber first scratched. “He has two kids. What is he, 26?” “Twenty-four, I think.”

I realize that I'm older than any three of them put together (giving new meaning to the title senior editor), but I don't bring it up. Nor do I bring up the observation that Jabber is, like its creators, still much closer to ground zero than to whatever it ultimately will become. In their “look, we can do this” quality, the presentations I saw at the show sent my mind back 20 years to when the PC was brand new, and everything you could do with it was novel and fun, yet still just prototypes for stuff that ultimately would become indispensable.

Like approximately everybody, I knew nothing about Jabber in early 1999. That was when my friend Perry Evans (who is perhaps best known as the founder of MapQuest) told me about it and wondered what I thought. He said Jabber was the quiet brainchild of a quiet 22-year-old programmer who hacked in the quiet farm country of Iowa. As instant messaging went, it had the same relation to AOL's system as the Internet had to, well, AOL. It was open source. It was XML. It let anybody deploy their own IM server just like they deploy an open-source web or mail server. And, because it was essentially an XML router, it had all kinds of interesting implications for just about everything.

Perry wanted to pump my brain for some thinking about how to support Jabber open-source development with some kind of business that would make money by selling Jabber-based products and services. I had ideas, but I knew Eric Raymond would have a lot more, so I invited him to join the conversation. Eventually Perry and his company, Webb Interactive Services, created Jabber, Inc., and I joined its advisory board (now rebranded the Open Board), as did Eric and other familiar folks from the Open Source community.

So here I was at Jabbercon, wearing three hats: *Linux Journal* editor, Jabber, Inc. Open Board member and speaker/panelist. When it came time to deliver the goods in that third capacity, I had to admit that after more than two years of trying to figure it out, nobody had the surefire formula for dealing with what Craig Burton, speaking at the same conference, called the “Infrastructure Paradox”, which is “balancing the business of generating shareholder value while fostering global ubiquity.”

But, there was some hope in his next remarks: “The market is constantly in the process of correcting itself toward construction modularity. This correction process causes alignment in architecture design and redistribution of competency.” Needless to say, this has been the story of Linux's success, especially as an embedded OS.

But it seemed to me that “redistribution of competency” is an overly commodified way of describing what was really going on here. Instead we had both obsoleting of irrelevance and origination of competency. What these guys were making with Jabber was not only highly original stuff, but stuff that was good for a lot more origination as well. Could it be that this was because these guys were not too far from having been originated themselves?

I don't think it's a coincidence that Linus Torvalds and Jeremie Miller were both around 21 when they introduced their inventions to the world, or that they've made it possible for the rest of us to do the maximum range of original work. The fact that both Linux and Jabber are still so young only makes them that much more indispensable.

email: doc@searls.com

Doc Searls is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*. His next book will be *Real Markets: What They Are and How They Work*.

Archive Index Issue Table of Contents

Advanced search

More on Trademarks

Lawrence Rosen

Issue #91, November 2001

Trademark selection is a job best done with the assistanceof your trademark attorney.

Selecting a good trademark, like selecting good wine, requires skill and sophistication. Intuition and common sense can often lead to ineffective trademarks, just as guessing over the wine list can lead to a disappointing dining experience.

Some marketing managers who should know better suggest legally ineffective trademarks for their products. Obviously I cannot reveal too many specific details, so I will create some examples to illustrate bad trademark choices I have seen. You cannot simply take a noun that is generic for the goods or services you're selling and make it into a trademark. This rule prevents someone from claiming private ownership over a common word. Thus, you can't use the word Computer as the trademark for a computer you sell.

You cannot use tricks of written or spoken language to get around these restrictions. Le Car would not be registrable to describe the newest model of automobile. Making the trademark sound like an exotic foreign language doesn't make the mark less generic. You cannot create a trademark simply by describing obvious product features and writing it in a unique way. Thus, PhoneRinger would not be registrable for a device that makes a sound when a telephone call arrives. And you cannot deliberately deceive consumers by being misdescriptive. The mark OpenSource could not be applied to a proprietary closed-source operating system. Trademarks that are “merely descriptive or deceptively misdescriptive” cannot be registered (15 U.S.C. §1052(e)).

Another frequent mistake is to try to capitalize on a competitor's trademarks (MacPizzas for a new chain of pizza parlors or Pentalium for a microprocessor) to get instant recognition in the marketplace. These trademarks probably would be attacked because they are likely “to cause confusion, or to cause

mistake, or to deceive" as to the origin of the goods or services (15 U.S.C. §1052(d)).

Trademarks can incorporate the adjectives Supreme, Superior or Enhanced, but such marks are of limited value. These terms are so over-used that they have lost their ability to distinguish goods or services. Also, you cannot take someone else's trademark and add one of those words to create your own trademark.

You must prevent your trademark or service mark from becoming generic for your goods or services. That is why Xerox tries diligently to prevent people from saying "I'm going to make a xerox of that document" instead of "Xerox copy". To prevent your trademark from becoming generic, always make sure it is used as an adjective and never as a noun.

A trademark does not exist in isolation. It must be used in conjunction with specific goods or services to signify the source or origin of those goods or services. For this reason, the same word can be used by more than one company as a trademark, as long as the goods or services are different. Thus, you can buy a Cadillac automobile and Cadillac dog food, and there will be no confusion among customers seeking one product and accidentally buying the other.

Trademark attorneys describe a spectrum of possible trademarks. In ascending order, which roughly reflects their eligibility for trademark status and the degree of protection accorded, these classes are:

Descriptive: certain marks are descriptive of the product, the name of the owner or the place where the product originates. They convey an immediate idea of the ingredients, qualities or characteristics of the goods. The distinction between a generic and a descriptive term was illuminated using the example Deep Bowl Spoon:

> "Deep Bowl" identifies a significant characteristic of the article. It is merely descriptive of the goods, because it informs one that they are deep in the bowl portion....It is not, however, the "common descriptive name" of the article, [since] the implement is not a deep bowl, it is a spoon...."Spoon" is not merely descriptive of the article —it identifies the article—[and therefore] the term is generic.
>
> —Fletcher, "Actual Confusion as to Incontestability of Descriptive Marks", 64 Trademark Rep. 252, 260 (1974)

Suggestive: a mark is suggestive if it requires imagination, thought and perception to reach a conclusion as to the nature of goods, for example, Orange Crush, Cuisinart and London Fog.

Arbitrary: when a common word is used in an unfamiliar way, examples include Apple, Apache and Python. Arbitrary terms often make good trademarks because they are likely to be more appealing to customers and are more easily remembered.

Fanciful: is usually applied to words invented solely for their use as trademarks, such as Altoids or Kodak. Fanciful words can make excellent trademarks because there is no possibility of confusion with real words.

Legal advice must be provided in the course of an attorney-client relationship specifically with reference to all the facts of a particular situation and to the law in your jurisdiction. Even though an attorney wrote this article, the information in this article must not be relied upon as a substitute for obtaining specific legal advice from a licensed attorney.

Please send legal questions regarding open-source software and related issues to info@linuxjournal.com.

email: lrosen@rosenlaw.com

Lawrence Rosen is an attorney in private practice in Redwood City, California (www.rosenlaw.com). He is also executive director and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (www.opensource.org).

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #91, November 2001

Readers sound off.

Free Enhydra?

The August 2001 *LJ*'s At the Forge is about a feature of Enhydra. Enhydra is commercial software, based on Java, which is commercial software. As far as I can see, there is no mention of Linux or anything specific to Linux in the article. It would not be out of place in a magazine called *Web Automation*, but I fail to see why it belongs in a magazine devoted to Linux, and more generally, to free software.

—Carl Fink

Reuven replies: While Java isn't free software, many web developers want or need to use Java in their work. I cannot change the Java license, but I can encourage Java developers to use an open-source operating system (Linux) and an open-source application server (Enhydra). So far as I understand it, there are two different ways to get Enhydra. The first is to obtain the core Enhydra application server, which is freely downloadable and distributable. This includes XMLC, DODS, super servlets and all that other stuff. The license agreement for this is indeed open source, despite what you might read elsewhere on the Enhydra web site. Here are three URLs about the licensing scheme: www.enhydra.org/aboutEnhydra/index.html, enhydra.enhydra.org/software/license/rationale.htm and enhydra.enhydra.org/software/license/opl.html. The second way is to get the Lutris EAS (Enhydra Application Server), which is a commercially packaged and tested version of the open-source stuff. I believe that you pay $995 US (or so) per CPU for EAS, but you're paying for the support, not for the server itself, in a twist on the open-source model. You could also argue that my article about PostgreSQL wasn't specifically about Linux, but the fact is that there are many people interested in using that database on Linux. For better or worse, there are lots of Linux devotees who also want to be able

to use Java. Indeed, as someone who does web development on Linux every day (and many nights) of the week, I'm happy to know that I can now talk to clients about a serious Java and EJB solution that goes above and beyond Jakarta-Tomcat.

Bug-Eaten Exploits

I really enjoyed the article “Loadable Kernel Module Exploits” by William Benton in the September 2001 issue of *Linux Journal*. I know that such articles are not meant to be exhaustive, but there is one bug in the Checking and Logging Function Listing on page 26 that I consider to be critical. The code hooks into the sys_call_table by saving the current address of SYS_write and replacing it. It unhooks by putting back the saved address. The bug is if there is more than one such hook. If another module hooks the SYS_write entry after this module, then an rmmod is done on this module, and the other module will become unhooked.

An even worse scenario is possible: 1) Module A is insmod-ed and hooks SYS_write. SYS_write points to wrapped_write in module A, orgwrite points to the previous contents (assume it is the normal entry). 2) Now module B does the same. SYS_write points to, say, modified_write in module B, orgwrite points to wrapped_write in module A. 3) Module A is rmmod-ed. SYS_write now contains the original SYS_write value. But this means that modified_write in module B is now bypassed. 4) Now module B is rmmod-ed. The cleanup_module in module B will restore the SYS_write table entry to point somewhere where module A used to reside. 5) Linux will most likely crash.

Unfortunately, I am not really a kernel hacker, so I don't know how I could tell the rmmod function that I don't want to be removed. To be safe, you would need to rmmod in the reverse order that you insmod-ed. I don't know enough about Linux to say how this could be accomplished.

—John McKown

Benton replies: You are correct; the situation you describe can be a problem for production code if a system administrator is not careful. I had not assumed that it would be a serious issue in example code (or in production code with a vigilant system administrator), but it is good to identify possible pitfalls in kernel programming. One (simple) solution is to be careful what you load and unload and in what order. As the old joke goes: “Doctor, it hurts when I do this!” “Well, don't do that, then!”. There are plenty of ways to crash your system poking around in the kernel, and it's best to avoid all of them once you've identified them. In all seriousness, these problems will exist as long as Linux does not provide a mechanism for registering changes to the system call table. I don't think they're horribly serious, since a degree of care on the part of the

user can prevent a crash. However, there is a workaround that can give you some basic stability: in cleanup_module(), be sure to check whether the system call table entry that you're about to restore is what you think it should be. If not, issue a warning and invoke the sync() system call before modifying the system call table. That will minimize the damage in the event of a crash. As an alternative (and more difficult) solution, you could replace the create_module system call in the first module you load, wrapping it with one that provides some sort of versioning system for the system call table in a static stack or deque. I'll leave implementing that as an exercise for the reader.

Kept Waiting

Please congratulate Lydia Kinata for the magazine's new design. I have been waiting a long time for this kind of improvement. Best regards and keep improving.

—Jorge Carminati

Change for the Better

I finally had time to read the September issue. The first thing I noticed was the new design. Looks great, keep the change! I've had a subscription for over two years and this, to me, was the best issue (look and content) I've received.

—Matt C.

Affordable Animation Apps

Robin, I got a copy of the *Linux Journal* at LinuxWorld and read your article on DreamWorks—what a great visit that must have been!

Given what you saw there and may already know of, I was wondering if you could offer a suggestion. I have a teenage son that is very interested in animation, but the applications to do that are pretty expensive. Can you suggest some Linux-based rendering applications that are more affordable?

—Jack C.

Robin replies: Jack, we had a great time at DreamWorks. But DreamWorks is just the first. All the studios have Linux migration projects underway.

You may want to start with Blender, a popular, freely distributed 3-D modeling package for animation, rendering, interactive 3-D and game creation. Unlike most no-cost animation tools, there are printed books available on Blender.

Other tools that may be of interest include K-3D, Ayam, AC3D, Flow, Radiance and BMRT. You can find their web pages by searching at Freshmeat.net

Remembering TSRs

In the September 2001 *LJ* article “Loadable Kernel Module Exploit”, I was interested in the technique for hijacking the sys_call_table pointer and wrapping it within your own function. I used to do something very similar in DOS in the good ol' days, writing TSRs (Terminate and Stay Resident) and other programs that “borrowed” the bios or dos interrupts vectors.

For programs, it was a simple case of restoring the original interrupt vector (pointer), but with TSRs, one had to be careful before restoring the original pointer. The problem being that if several TSR utilities were installed, two might hijack the same interrupt (often the keyboard). A properly written TSR would, before uninstalling, check that the current pointer pointed indeed to itself, if it didn't, then it could not uninstall as by doing so it would clobber another, later installed, TSR.

The cleanup_module(void) routine in the article performs no such check before restoring the sys_call_table[SYS_write] pointer, something which at first sight appears dangerous to me in a multitasking environment, especially one where the kernel can load and unload modules automatically.

—John Jacq

Archive Index Issue Table of Contents

Advanced search

UpFront

Doc Searls

Issue #91, November 2001

Stop the Presses, *LJ* Index and more.

Linux Bytes Other Markets

Two years ago I was working as VP of R&D for a large telemarketing company. My job involved building and managing very large databases, primarily for the purpose of target marketing. We were spending an arm and a leg on software licenses in the R&D shop and another arm and leg on additional software elsewhere in the company. Although we did some things well, our expensive software was still not providing managers with the information necessary to make sound business decisions. For example, we were too late with information on profitable and nonprofitable campaigns, and we never did have good data on cash flow. I left that company after our R&D office was closed, disappointed that some good work went down the drain. I was also frustrated by the lack of software solutions for the company's basic business problems—and with just a hint of understanding of the board's unhappiness with how much our software licenses were costing.

After a few months of private consulting in database design, I was invited to a job interview by the CEO of a medium-sized manufacturing company named Action Target. (Action Target is in the business of manufacturing and installing target equipment and shooting ranges.) I was shocked at what I discovered. The CEO (an electrical engineer with an entrepreneurial bent) had the whole place running on Linux. Every single user, from sales, shipping and purchasing to production, engineering and accounting, worked on a Linux workstation, most of which were diskless. These workstations were served from a group of servers, which included a web server, a database server and collection of application servers. The application servers hosted all home directories in addition to the applications. The applications were mostly of the open-source or free variety, such as StarOffice, Netscape and TGIF, but also included a few proprietary applications, like Applix. The database server hosted PostgreSQL.

Networking was accomplished via NFS between the servers, and the diskless machines were networked via bootp to the application servers.

What I discovered next shocked me even more. Not only was the whole place running Linux, but the CEO had written his own ERP (enterprise resource planning) application. He told me that he had investigated several other solutions, but they were all expensive, ran on expensive and proprietary RDBMSes and, most importantly, did not permit full customization. Therefore, he explained, he decided to write his own. The front-end tool he chose was Wylib, which he had coded. Wylib is a group of libraries written using Tcl/Tk. Tcl/Tk is easy to use, platform-independent, can act as a shell and belongs to and is widely accepted in the Open Source community. For the back-end RDBMS, he selected PostgreSQL.

This customized ERP system runs the company. Employees use the system because they can't do their jobs without it. The ERP is based on open-source software and is fully customizable. In order words, if we're not happy with it, we change it. Management estimates that this system has saved the company over $1 million (US) in operation expenses, not to mention the huge savings in software licensing.

After working at this company for a year, recoding the ERP using Wylib and restructuring the database, I've been spun off into my own company, WyattERP. WyattERP's mission is twofold: 1) to show companies how to save money by using open-source software and 2) to show companies how to create an ERP solution using Wylib that is inexpensive and completely suited to the needs of the business.

Wylib and the sample code can be downloaded from the web site, www.wyatt-erp.com. Wylib is open source and distributed under the OPL (Open Public License) agreement. I can be contacted via e-mail through the web site.

—Merrill Oveson

Optimistic Signs for Embedded Linux

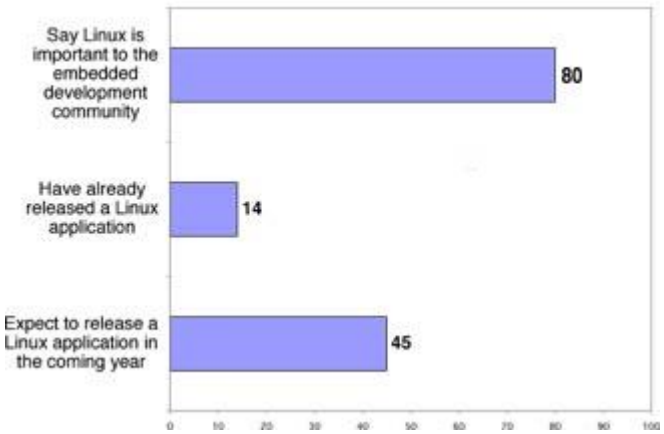
It's no news that Linux is finding a home in everything “from household appliances and consumer electronics gadgets to controllers used in aviation, factories and automobiles—application-specific devices”, as Evans Data Corporation (www.evansdata.com) reports. But it is news to discover how high the Linux embedding rate appears to be among those many devices.

In its “Embedded Systems Developer Survey”, Evans Data reports what it calls “a dramatic change in the software content of the microprocessor-controlled devices in the workplace and in our homes”. As a result, the firm predicts a

triple increase in Linux-based projects over the next year, which it says is “part of a shift toward commercially available operating systems rather than those developed in-house.”

Tools are an issue. Without giving numbers, the study showed “keen interest in tools and products that can help developers get their projects completed correctly and on time”. Two-thirds of the surveyed developers said many embedded projects are started and never completed, and many more are delivered late.

What is it that developers like about Linux? Just what you'd expect: “Open-source code, royalty-free licensing and a large community of knowledgeable developers were cited as Linux's key benefits.” On the down side, Evans Data reports, “Factors slowing Linux adoption include lack of specific board support packages, availability of device drivers and fragmentation of the code base.”



—Doc Searls

LJ Index—November 2001

1. Percentage of early-adopting PVR (personal video recorder, like the Linux-platformed TiVo) owners who are watching more TV than before they owned the device: 63
2. Percentage of current PVR owners who have “no idea” which channels the shows being watched were originally on: 12
3. Percentage of current PVR and VCR owners who like to scan past ads: 25
4. Number of channels on Dish Network's $31.95-per-month Dish 150 plan: 215
5. Number of open-source licenses on the Open Source Initiative's approved list: 22
6. Percentage of 74 leading brands that have lost brand value over the last ten years: 55

7. Percentage drop in value for all 74 leading brands over the same period: 5
8. Number of new products introduced last year: 31,432
9. Royalty charge paid by recording equipment manufacturers, according to the American Home Recording Act (AHRA), as a percentage of the manufacturer's revenue: 2
10. The Sound Recordings Fund's share of AHRA royalty payments: 2/3
11. Percentage of that fund allocated for “non-featured musicians and vocalists”: 4
12. Percentage of remaining royalties allocated for “featured recording artists”: 40
13. Percentage of remaining royalties allocated to “owners of the exclusive right to reproduce sound recordings distributed during the relevant year”: 60
14. Red Hat percentage on machines surveyed by Linux Counter: 28.39
15. Slackware percentage on machines surveyed by Linux Counter: 22.40
16. Debian percentage on machines surveyed by Linux Counter: 19.30

Sources

1-3: Electronic Media, from NextResearch

3: *The Wall Street Journal*

4: Dish Network

5: Open Source Initiative (www.opensource.org)

6-8: *Financial Times*

9-13: RIAA

New Software Project

Jan Schaumann has begun a new software project that will serve a dual purpose: 1) to produce MakeMan, a project that will provide several GUI and non-GUI front ends to an XML interface for writing man pages and 2) to document the steps of an open-source project, from registering at SourceForge to releasing packages in great detail. See mama.sourceforge.net for details.

They Said It

Show me a web app that can't be served from a Pentium 100 and I'll show you a dead dot-com.

—Kevin Jamieson

If you want to be a platform vendor, and you want developers to invest alongside of you, your platform needs to be open source. This in fact has been true for the last few years.

—Dave Winer

All power is derived from the barrel of a gnu.

—Mao Tse Stallman (from the .sig of Alan Olson)

Technological progress is like an axe in the hands of a pathological criminal.

—Albert Einstein

They're the next dead thing.

—Grady Hannah of Linuxcare on a company other than his own that he'd rather leave nameless.

The operating system wars are over, and operating systems lost. What we are now witnessing is the growth of environments where applications span multiple operating systems, rather than vice versa. Individual computers are the new object, with a handful of exposed interfaces and complicated internals hidden from the caller by standard protocols.

—Clay Shirky

The real problem we face with the Web is not understanding the anomalies, it's facing how deeply weird the ordinary is.

—David Weinberger

This is an era when nonsense has become acceptable and sanity is controversial. Too many people fail to see a distinction between “the rule of law” and the edicts of judges. Unfortunately, these people include many judges.

—Thomas Sowell, in 1996

One can never consent to creep when one feels an impulse to soar.

—Helen Keller

Today we turn our dagger around and point it at ourselves. As goes the Internet Economy, so goes the magazine founded to cover it.

—Jimmy Guterman, uttering the parting words of *Media Grok*, the e-mail newsletter of *The Industry Standard* on the day it went out of business.

What lever did the invisible hand have to throw to start up the ghost in the machine?

—Christopher Locke

Stop the Presses: HP Compresses Compaq

The news caught everyone by surprise. Hewlett-Packard was buying Compaq for $25 billion (US) in stock. After both stocks were hit the next day, the deal was worth about $20 billion—even though the announcement boasted that the combined company "Will Have Number One Worldwide Positions In Servers, PCs and Hand-helds, and Imaging and Printing; Leading Positions In IT Services, Storage, Management Software."

The numbers involved were staggering. Combined revenues exceeded $87 billion over the past four quarters, which would place the new HP second only to IBM ($90 billion) in size. The new behemoth would also have 145,000 employees in 160 countries—prior to staff cuts, which are sure to come mostly at Compaq's expense. If the deal goes through, Compaq as a company will only persist, in the manner of Netscape after AOL, as a "brand". Together HP and Compaq brands account for about 70% of PC sales through retail outlets; but Dell now leads in sales overall, having moved ahead of both HP and Compaq. This, pundits mostly agreed, was clearly one strong motive for the deal.

Reviews in the mainstream media were mixed at best. Dan Gillmor of the *San Jose Mercury News* wrote, "It's hardly thrilling to be the leader in a market that is dull, nearly devoid of innovation and barely profitable." In the UK, *The Register* wrote:

> A braver choice for HP would have been to prise its way into new infrastructure partnerships, swallowing a Nokia or a Nortel. Or even an ARM. But instead of looking forward, HP has looked back and fallen on an acquisition target that looks agreeably like itself only financially weaker.

But in the Linux community, the mood was more upbeat. Linuxcare cofounder Dave Sifry said, "Gee, everyone said that there'd be consolidation in the Linux space, but this is a bit bigger than I expected!"

Dan Kusnetsky, VP System Software for International Data Corp. and one of the leading Linux analysts, says:

> In the recent past both companies have been looking for ways to lower their overall software development costs by moving to high-volume, third-party software. This is quite reasonable. It is quite expensive to be a world-class supplier of operating systems.

He adds that the combined company would be supporting as many as eight different OSes, and offers these predictions: 1) since Tru64 UNIX is the lower-volume product and is tied to Alpha, the best features of Tru64 UNIX will be merged into HP-UX and Linux, and then Tru64 UNIX will be retired; 2) the best features of OpenVMS and MPE/ix will be merged into Windows and then one or both of those operating systems will be retired; and 3) the combined company will focus on HP-UX for high-end enterprise tasks, Windows for workgroup and desktop tasks, and Linux for Web-centric computing tasks.

Which isn't exactly bad news for Linux.

—Doc Searls

Archive Index Issue Table of Contents

Advanced search

Enterprise: Linux, Starships and Postscripts

Richard Vernon

Issue #91, November 2001

Ready or not, here it comes.

Always playing with the semantics of each issue's focus, Marcel Gagné reminds us that for many, the word enterprise conjures up visions of Captain Kirk and the starship under his command. But by Captain Kirk's time, the name *Enterprise* (sometimes *Enterprize*) was already an old one for warships. A 12-gun US naval schooner built in 1799, the third US naval vessel to bear the name, is perhaps the most famous (next to the starship, of course). It was this little ship that spawned the tradition of passing the famous name to innovative or "enterprising" ships in the 20th century (i.e., a WWII aircraft carrier and the first nuclear-powered aircraft carrier). The schooner served with distinction through the engagements with the Barbary corsairs, the Quasi War with France and the War of 1812. She was commanded by some of the most notable names in US naval history, such as Stephan Decatur, Isaac Hull and James Lawrence (of "Don't give up the ship" fame). Michael Bosworth writes that "*Enterprize*, in her original schooner rig, had a reputation of being fast, and she was remarkably lucky in her assignments, her timely refits and her officers and crew." When the time came in 1803 to build more naval schooners, it was hoped that the *Enterprize* could serve as a model. But alas, no *Enterprize* plans were to be found, and the schooner, while lucky in her own right, could not be reproduced. Her career was a collection of serendipitous and fortuitous events that entice one to believe she was fated to succeed.

Despite the hard work of thousands, the point where Linux now sits can similarly be considered a lucky or fated enterprise, especially when one considers the role initially played in that success simply by the personality of Linus Torvalds. But now Linux seems to be at a juncture. The initial hype and excitement are gone, the bandwagon jumpers-on are gone, and it's time to see whether Linux can continue its maturation rate and present a viable solution to an ever-increasing variety of applications.

While it's important not to forget the many improvements yet to be made to Linux, I like the attitude in David Bandel's column this month—that the existence of many large companies using Linux is proof that it is ready for the enterprise. Our article on Bike Friday and their move from Microsoft to Linux supports this, and this month's Linux Bytes Other Markets highlighting a manufacturing firm that uses Linux for everything, including every employee's desktop, is another drop in the bucket of evidence that this player is ready for prime time.

As a side note, plans for a ship believed to be the schooner *Enterprise* were recently found in Venice. The plans are being used to construct a replica in Washington, DC that will be used as an educational ship and goodwill ambassador.

email: info@linuxjournal.com

Richard Vernon is editor in chief of *Linux Journal*. He enjoys studying naval history and really knows his way around a dinghy.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #91, November 2001

Our experts answer your technical questions.

Mounting Mac OS Partitions

I have a 2001 iMacDV (400MHz, 128MB RAM) running Mac OS 9.1 and Yellow Dog 2.0. I cannot gain access to files on my Mac OS 9.1 partition. I assumed that my Mac OS partition was hda1, but this may be wrong.

—Bill MacKay, w.mackay1@ntlworld.com

It's not hda1. Macs use a rather complex partitioning scheme. Use the command **cat /proc/partitions** to see all your partitions. That should help you narrow it down.

—Ben Ford, ben@kalifornia.com

Firewall Takes Hours to Start

I upgraded the kernel of my Red Hat 6.0 system to 2.2.16-3 in order to use ipchains. I used the instructions from www.redhat.com support to upgrade the kernel. It now works great as a dedicated firewall. Whenever the power goes out, I reboot the machine and run the /etc/rc.d/rc.firewall script, which I obtained from www.linuxdoc.org, and this takes an hour to complete. Is there some way to speed up this process?

—Paul Lamping, palamping@yahoo.com

Are you using a large hard drive with ext2 as your filesystem? In that case, the process that is taking so long is the filesystem checking process due to an improper shutdown. There are a couple of things you can do about that. You can try a journaling filesystem, use a UPS battery backup to shut down

gracefully, or get creative with your drive partitioning and make as much of it read-only as you can.

—Ben Ford, ben@kalifornia.com

Setting up your firewall rules should be a matter of seconds, not minutes or hours. One thing that may be causing problems is if you use names for hosts rather than IP addresses. If your boot scripts start your name server after your firewall script runs, this forces each line that does a DNS lookup to time out before continuing. This can also be the case if you refer to a name server on another machine and your firewall script locks the machine out before creating "allow" entries.

—Chad Robinson, crobinson@rfgonline.com

Printing Stopped Working

I have been printing to a Jetdirect-enabled HP LaserJet 4Plus on our LAN for several years using its IP address. It recently stopped working. Running any lp command (lpq, lpr, etc.) returns the following message:

```
Printer 'lp@localhost' - cannot open
connection
Connection refused
Make sure LPD server is running on the server
```

—Murray Zangen, murray@nj.com

Have you tried rebooting the printer? This problem is a symptom of the recently seen CodeRed IIS and indexing server worm. It stops many printers and Cisco devices cold.

—Ben Ford, ben@kalifornia.com

It appears that your problem is local to the box, not to the HP printer. LP is telling you that LPD is not running on the local server, which it must connect to in order to queue the request for the remote printer. Check the output of **ps ax** to ensure that LPD is running properly. If it is, you should also check connectivity to your printer manually by Telneting to the IP address given by /etc/printcap. Finally, make sure you are selecting the correct print queue, and print a test job using **lp** from the command line rather than printing from an application.

—Chad Robinson, crobinson@rfgonline.com

How to Change Samba Passwords

I have configured Samba and my server shows up in Network Neighborhood on my Win98 workstations. I can see the folders on my server, yet it asks for a password to access the folders. I have tried resetting the password in Samba for that shared folder, but I can't seem to figure out where the right place is to set it.

—Dan Schmeh, dschmeh@turningpnt.org

The standard installation uses /etc/passwd. If you use the standard installation you need to modify the registry on the browsing machine to allow for unencrypted passwords. Add the following key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services \VxD\Vnetsup\EnablePlainTextPassword = 1
```

and reboot. There is a good discussion of encrypted vs. un-encrypted passwords in the Samba distribution. On Red Hat 7.0, the file is /usr/share/doc/samba-2.0.7/docs/textdocs/ENCRYPTION.txt.

—Christopher Wingert, cwingert@qualcomm.com

How to Start a Web Server Automatically

httpd doesn't automatically start during boot time. I have to start it manually from the /etc/rc.d/init.d/ directory. Which config file should I modify to make it start at boot time?

—W. Huang, whuang53@excite.com

Run **chkconfig --level 5 httpd on**. This assumes you run in runlevel 5 (GUI in Red Hat). If you want other runlevels, just change the 5 to the number of the runlevel you wish to change.

—Ben Ford, ben@kalifornia.com

I Have No Phone and I Must Connect

I live a few miles off the electric and phone lines with no line of sight to a repeater dish in one of the largest and poorest counties in the US.

For those of us with remote homes in the mountains near the Canadian border who have gotten computers during the last few years, internet access is many miles away.

While DirecPC works with a satellite connect down, one needs a telephone line out to the Net. Many of us do not have phone access, yet a radio system less expensive than the Ham radio would make a great change in our lives. Our download needs are greater than our upload needs. How do we make this work?

—Robert Thomas, homeschoolu@hotmail.com

You can run DirecPC from Linux with a router from www.helius.com. They are scheduled to come out with a two-way router for bidirectional DirecPC in the third quarter of 2001. My suggestion to you would to be set up a network with your neighbors with a caching server to speed access.

—Ben Ford, ben@kalifornia.com

Community net groups are using standard WiFi cards and modified satellite dishes to make high-speed connections that span miles. You might be interested in using this technology to share your satellite access with neighbors. See www.toaster.net/wireless/community.html.

—Don Marti, info@linuxjournal.com

Adding a Hard Drive

I installed and partitioned a second hard drive, but I can't make a filesystem on it.

—Kevin Williams, williams_kevin@btconnect.com

First, figure out what device you are talking about. Here is a simple chart to determine device names: /dev/hda1 master drive on primary channel, 1st partition /dev/hda2 master drive on primary channel, 2nd partition /dev/hdb1 slave drive on primary channel, 1st partition /dev/hdb2 slave drive on primary channel, 2nd partition /dev/hdc1 master drive on secondary channel, 1st partition /dev/hdc2 master drive on secondary channel, 2nd partition /dev/hdd1 slave drive on secondary channel, 1st partition /dev/hdd2 slave drive on secondary channel, 2nd partition Next, choose the filesystem you would like on it. SuSE includes reiserfs with their distro; I recommend that as it doesn't require tedious filesystem checks on a bad reboot. Then format the drive. The command used to make reiserfs partition is **mkreiserfs <*devicename*>**, where <devicename> is the device name of your partition. If you want to stick with ext2, use the same command but replace mkreiserfs with mke2fs. Now you have a useable drive partition; all that is left is to mount it. Choose or create a mountpoint in your filesystem. I will use /mnt/storage for this example. Create

the mountpoint with **mkdir /mnt/storage**. As you see, the mountpoint is nothing more than a directory. Now mount the drive like this:

```
mount <devicename> /mnt/storage -t <filesystem
type>
```

where <devicename> is the device name of the partition and <*filesystem type*> is the type of filesystem. You now have another useable hard drive. One step remains. I assume you want this accessible next time you boot the system, so we need to add the partition to /etc/fstab. Add a line like this to that file:

```
<devicename>     <mountpoint>     <filesystem type> defaults     0 0
```

where <devicename> is the device name of your partition, <*mountpoint*> is the mountpoint you are using and <*filesystem type*> is the type of filesystem on that partition. If you choose to use reiserfs, leave the last two numbers as 0 0. If you use ext2, make those numbers 1 2.

—Ben Ford, ben@kalifornia.com

Can't ping

The installation of Red Hat 6.2 on my Dell OptiPlex GX150 went well. However, after the system came up, I couldn't ping any host, even on the same subnet. I typed **arp -a** and the system hung. The system has an integrated 3Com 920 10/100 BT card.

—Khoa Nguyen, knguyen@megisto.com

Use netconfig to set up your networking. You can also use lsmod to see if the module for that card is loaded. I would guess that the module for that card would be the 3c90x. You may have to load this module by hand if the card isn't automatically detected by using the command **modprobe 3c90x**. If this works, edit the file /etc/modules.conf and add the line **alias eth0 3c90x**.

—Ben Ford, ben@kalifornia.com

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #91, November 2001

PDG Shopping Cart 2002, PitBull LX, Network SecurityServices and more.

PDG Shopping Cart 2002

Designed for small- and medium-sized businesses, Shopping Cart 2002 is a complete e-commerce shopping cart that requires no custom coding and features a web administrator interface, affiliate tracking and on-line credit-card authorization. User-defined product data fields, the StoreBuilder template, support for Boolean searches, multitiered pricing, IP tracking, on-screen help and real-time price calculation are among the new additions to the 2002 version. Security measures include SSL compatibility for 128-bit encryption, GPG encryption and the ability to exclude payment data from the order log and e-mail notifications. A free 30-day evaluation of Shopping Cart 2002 is available on the PDG Software web site.

Contact: PDG Software, Inc., 1751 Montréal Circle, Suite B, Tucker, Georgia 30084, 770-270-0062, presales@pdgsoft.com, www.pdgsoft.com.

PitBull LX

PitBull LX, application security software from Argus Systems Group, is now available for many popular Linux distributions (SuSE, Red Hat, Debian/GNU, etc.) as well as the 2.4 kernel. PitBull LX protects against security flaws by isolating applications in separate security compartments, thereby containing bugs to a single compartment rather than allowing system-wide access. Other features include domain-based access control for user, file, process and network; root restrictions; protection against known and unknown bugs and worms; and a 30-minute installation process.

Contact: Argus Systems Group, Inc., 1809 Woodfield Drive, Savoy, Illinois 61874, 217-355-6308, info@argus-systems.com, www.argussystems.com.

Network Security Services

Network Security Services (NSS) is a set of libraries and tools designed to support cross-platform development of cryptographic applications and security-enabled network applications. The NSS libraries export a C API that C or C++ applications can invoke to perform crypto operations, handle certificates, send and receive S/MIME messages, and communicate securely over the network using SSL and/or TLS. NSS supports SSL v.2 and v.3; TLS; PKCS numbers 5, 7, 11 and 12; x. 509 v.3 certificates; and other security standards, as well as hardware crypto accelerators and smart cards. Source code and binary distributions are available for download.

Contact: Network Security Services, The Mozilla Organization, www.mozilla.org/projects/security/pki/ns.

Zend Accelerator

Zend Technologies announces version 2.0 of Accelerator, its server-side caching program designed to improve the response time of PHP dynamic scripts. Accelerator 2.0 features a module that enables system administrators to measure real-time site acceleration, multipass optimization with the CGI functionality of PHP 4 and a quick installation. As a result of code parallelization, Accelerator increases the number of requests handled per second by a factor of three. Latency time is also trimmed to near zero. In addition, memory-intensive scripts rarely visited can be excluded.

Contact: Zend Technologies, Inc., 11 Penn Plaza, 5th floor, Suite 5013, New York, New York 10001, 877-936-3872 (toll-free), info@zend.com, www.zend.com.

HP Workstations x2000, x4000

HP has made available Linux-based versions of its x2000 and x4000 workstations. Workstation x2000 is based on Intel's 850 chipset Pentium 4 processor and clocks up to 1.7GHz, while workstation x4000 is based on the 860 chipset and can include one or two Xeon processors. Both systems provide a range of 2-D and 3-D applications with guaranteed ISV certification. The x2000 and x4000 workstations are geared toward graphics and memory-intensive technical applications, such as digital content creation and animation. Both systems ship with Red Hat 7.1, and full technical support is provided. Full features and optional components of both can be viewed at www.hp.com/workstations/products/linux.

Contact: Hewlett-Packard, 3000 Hanover Street, Palo Alto, California 94304, 800-752-0900 (toll-free), www.hp.com.

Tarantella Enterprise 3 Starter

Tarantella, Inc. released Tarantella Enterprise 3 Starter, network infrastructure software designed for small enterprise or departmental use. It can be used for a range of tasks, from remote system administration, server support and application access, to integration with servers running on other platforms for access to all corporate applications. Other uses include integration of web-based applications into an existing network and a drop-in wireless gateway. Tarantella Enterprise 3 Starter is available for a variety of distributions, and an evaluation copy can be downloaded from their site.

Contact: Tarantella, Inc., 425 Encinal Street, Santa Cruz, California 95061, 888-831-9700 (toll-free), www.tarantella.com.

NetCam

StarDot Technologies announces NetCam, a standalone, IP-addressable, internet camera with an integrated web server. The built-in 10-BaseT Ethernet connection allows NetCam to plug directly into a LAN, cable modem or DSL network, allowing remote access via any web browser without additional plugins. Running on µClinux and a ColdFire CPU, it comes with 8MB DRAM and 2MB Flash memory, and supports TCP/IP, HTTP, FTP, ARP, Telnet and Daytime protocols. Image resolution is 640 × 480 RGB, and images can be archived automatically on a remote server. NetCam weighs 19.5 ounces and has dimensions of 3.25" × 2.20" × 6.56".

Contact: StarDot Technologies, 6820-H Orangethorpe Avenue, Buena Park, California 90620, 714-228-9282, stardot-tech.com.

Archive Index Issue Table of Contents

Advanced search