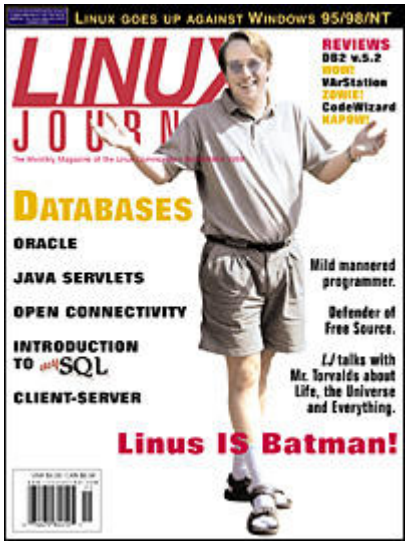
# *Linux Journal* Issue #67/November 1999




***Focus***

Databases *by Marjorie Richardson*

***Features***

Creating a Client-Server Database System with Windows 95 and Linux *by Liu Kwong Ip*

Here's the way to develop a dial-on-demand database cluster in Linux.

Using Java Servlets with Database Connectivity *by Bruce McDonald*

The persistent nature of Java servlets makes them ideal for database/web technology. Mr. McDonald takes a look at using servlets with PostgreSQL and JDBC.

Open Database Connectivity *by Peter Harvey*

Mr. Harvey describes the ODBC open specification for application developers.

MySQL Introduction *by David Axmark and Michael Widenius*

A look at the MySQL database—where it's been, where it is now, and where it's going.

Oracle Database Administration on Linux with ORAC-DBA *by Kevin Kitts*

Database Administration got you down? Here's a tool to help you out.

***Forum***

Performance Comparison *by Tim Newman*

A look at computational performance for an appoication running on an x86 CPU with Linux and Windows 95/98/NT, and how they compare.

PHP version 4 *by Craig Knudsen*

This personal tool to a popular Apache module is now in version 4 beta.

Linus Torvalds *by Marjorie Richardson*

A conversation with the man himself.

### *Reviews*

ICP vortex GDT RAID Controllers *by Eric Green*
DB2 v5.2 for Linux *by John Kacur*
CodeWizard for Linux *by Ben Crowder*
VA Linux Workstation VArStation XMP *by Jason Kroll*
MySQL and mSQL *by Reuven M. Lerner*

### *Columns*

**System Administration** EMU—Event Management Utility *by Jarra Voleynik and Anna Voleynik*

The authors present a freely available tool for monitoring enterprise systems through simplicity toward complexity.

**At the Forge** Working with LWP *by Reuven M. Lerner*

**Focus on Software** Focus on Software *by David A. Bandel*

Description

### *Departments*

Letters

More Letters

upFRONT
Penguin's Progress: Hacking an Industry *by Doc Searls*
Best of Technical Support
New Products

### *Strictly On-Line*

NoSQL Tutorial *by Giuseppe Paterno*

A comprehensive look at the NoSQL database.

Interfacing Relational Databases to the Web *by Will Benton*

This document explains how to build a database-backed web site using Apache, the php3 module and the PostgreSQL relational database.

Automating IP Host Data Collection on a LAN *by Joe Nasal*

Using Linux and the ACEDB database system makes easy work of local area network management.

Buy One, Get One Free *by R. Scott Gray, Luke Pargiter and Phil Pfeiffer*

Configuring Linux as a dual-purpose user environment.

Linux in the Tropics *by Rodrigo Bernardo Pimentel*

Linux is flourishing in Brazil. There is a Brazilian distribution, several high-traffic mailing lists, articles on mainstream

newspapers and many users. Now, they have their first Linux User's Association.

Mastering Linux *by Bob van der Poel*

Linux Configuration & Installation, 4th Ed. *by Bob van der Poel*

A book review.

How to Install and Configure Oracle on Linux *by Greg Flannery*

A step by step demonstration of the Oracle installation process.

Archive Index

Advanced search

# Focus: Databases

**Marjorie Richardson**

Issue #67, November 1999

If you don't learn something about databases this month, you must be an expert on them already.

What is there to say about databases? They are one of the most important tools in enterprise, and yet at first glance they seem boring. However, authors and users both love them, as can plainly be seen by the number of articles (ten) that were submitted for this issue. We present five here, three in Strictly On-line and put off two for next month. We'll look at MySQL, NoSQL, PostgreSQL, Progress (next month), Oracle and the ODBC (open database connectivity) project, as well as using Java and the Web with databases. There are also reviews of DB2 and a book on mSQL and MySQL. If you don't learn something about databases this month, you must be an expert on them already.

## Our Apologies

Last month, we printed an article, “Multilink PPP: One Big Virtual WAN Pipe” purportedly by Paul Ammann. We have since learned the article was actually written by George E. Conant, formerly of Xyplex, and can be found on the Web at www.data.com/tutorials/multilink_ppp.html. We have not been able to reach Mr. Conant to offer our apologies personally, but do so here in this forum. We put a large amount of trust in the authors with whom we contract; Mr. Ammann has betrayed this trust both to us and to our readers. When I asked him about it by e-mail, he sent this reply:

```
From: pammann@tekrab.net (Paul Amman)
Subject: RE: [info@linuxjournal.com.com: Boo! Hiss! (fwd)]
Date: Tue, 31 Aug 1999 20:49:05 -0400
Well... to be honest, yes, I did copy the article.
I guess there's no hiding it and it's
time to own up to it now. There is no good
explanation that would be acceptable for my
behavior. Please advise me on what retribution I
must make. I will accept full responsibility for
my actions.
Paul
```

We are extremely distressed that this happened—the only time in over five years of publishing.

—Marjorie Richardson

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL



# Creating a Client-Server Database System with Windows 95 and Linux

**Liu Kwong Ip**

Issue #67, November 1999

Here's the way to develop a dial-on-demand database cluster in Linux.

About half a year ago, we began a project called NORA to develop an information system for a private dental clinic in Hong Kong. The basic requirement was that the clinical information, including patient folders, appointment books, laboratory work, etc., could be retrieved and edited by any client PC in the clinic. In addition, the users hoped they could access the data from another clinic using the same system. The system is now in beta testing. We gained some valuable experience during this project, which may be useful for someone wishing to develop a similar system, especially for small- to medium-sized businesses.

We established the following requirements:

- a client-server database system
- multi-site (de-centralized)
- connectivity between LANs on demand
- dial-up service
- Windows 95 client
- Big5 Character set support
- low transaction rate
- portability

Since the system would be needed by several users (a dentist and nurses) at the same time, a client-server system is expected. The users do not know much about computers, but they do know Microsoft. They insist on using Windows 95 as the operating system of the client PC, so that they can use their favorite office suite with the same machine. For the server part, they have no

preference, so we could decide. We considered both Windows NT and Linux. After considering the stability, ease of installation, cost, flexibility and the requirements listed above, we chose Linux. We think we made the right choice; otherwise, the other system requirements could not be easily fulfilled.

The system is used by a group of several clinics. Users wished to retrieve and update all data easily from any clinic. We considered implementing the system on a single big server, with all clinics connected to it by telephone line or ISDN. However, we found that not only are the communication costs and efficiency worse than the decentralized system (i.e., each clinic has its own server), but also much work would be necessary if another clinic joined the centralized system, since the data in both databases would need to be merged.

As we chose to have one server in each clinic, the connections between clinics should be made on demand, i.e., the connection should be established only when needed. We could install a modem for each client, so that one could dial the server of another clinic to access the data independently. However, this is not an effective method, since every client machine would need a modem and telephone line, and most of the time they are idle. We proposed that the connection be established by the servers, and the clients access the data at another clinic through the servers on demand.

We tried to use **diald** (dial daemon) on Linux to provide this function. However, most of the documentation on diald assumes the user is using it on a stand-alone workstation or to dial an ISP to access the Internet. The consequence is that the connection is not two-way, i.e., the machine on the Internet cannot access the local workstations. Moreover, the configuration in the document did not consider having dial-in service on the same machine, and the two kinds of service may or may not be compatible.

We found a way to configure diald and dial-in service on Linux harmoniously. Thus, all machines in one clinic can access the database server in another clinic based on dial-on-demand, while the machines in the other clinic can access the database in the first clinic at the same time.

Dial-in service is needed for the dial-in request from the server in another clinic to connect the two LANs. Moreover, the users want to access the data, even when they are at home, by a stand-alone Windows 95 workstation with a modem.

As mentioned above, the users insist on using Windows 95 as their front end. Finding proper method for connecting the client software in Windows 95 to display the data in the Linux server was a problem. This is because some of the database servers for Linux do not provide this feature.

Another constraint for the server is that it must support Big5 characters, since most of the patients use their Chinese name and address for registration. This almost forces us to the final choice of database server, the MySQL server.

We estimated the transaction rate of the system server and found it should be relatively low, about ten SQL executions per minute in the peak period. We think this property is common for small- to medium-sized business applications, so the loading performance of the database server is not crucial.

Finally, we always kept portability in mind. Even though the client software is implemented on Windows 95, we hope to port it to another platform in the future.

### Database Server Consideration

When this article was being written, Informix-SE for Linux was just becoming available, and Oracle had started to port their database server to Linux. We did not consider these two popular databases. We considered only database servers with these basic properties:

- It follows the relational model and supports SQL (or a subset of SQL). The relational model has become a standard for modern database servers. Our client software can communicate with the database in the standard way using SQL. Therefore, even if we change the database server software in the future, most of our client code will not need to be changed.
- It is free or low cost. We believe one can find good software for free or low cost in the Linux world so we considered the free database servers first. If none had satisfied our requirements, we could then have considered a commercial one.
- It is open source, if possible. We wanted the source code of our whole system including operating system, client-server software and database server, so that our system would not be affected by any standard or format changes by third parties. Of course, we cannot have the code to Windows 95, which is why we want to port the client software to another platform.

The database servers were compared in four aspects: available C API, available ODBC driver, Big5 code support and concurrency control method. C API is important for the client software running on Linux. The ODBC driver is for client software running on Windows 95. Big5 code support, as mentioned above, is one of the basic requirements of our system. Since it is a multi-client system, the method for preventing concurrent client data access from interfering with each other is also important. We carried out a survey of four popular database systems: PostgreSQL, Beagle SQL, mSQL and MySQL. All source code to these

systems is available. Both PostgreSQL and Beagle SQL are free of charge. mSQL is free if you use it in academic and registered charity organizations, otherwise it costs $250 US for a single license. MySQL is free if you don't sell it; otherwise, it costs $200 US per copy. The results are shown in Table1.

Table 1.

MySQL was chosen as our database server. The most important reasons were that it has an ODBC driver for Windows 95, and it supports Big5 characters. MySQL is a multi-threaded process, with one thread for each connection. Moreover, many support utilities such as table repairing tools are provided. We recompiled it for Big5 support. During beta testing, we found the system to be stable, efficient and reliable on Red Hat 4.2. However, we found it could not successfully compile and run on Red Hat 5.0, even when we strictly followed the manual. We think the main reason is library incompatibility.

## Client-Side Consideration

We considered different C++ (or C) compilers for Windows 95, and finally chose Borland (Inprise) C++Builder as our client-side software development environment. Some visual objects are in C++Builder (similar to Delphi and Visual Basic) to access the content of a table in the database directly. They are supposed to be simple and easy to use. However, when the program becomes large, maintenance of the code with these kinds of objects is not easy, because the behavior of each database widget is almost independent. We decided to develop a layer of database objects to act as a bridge between the visual objects and the database content. The SQL statements are embedded in the database objects. In this way, the clinical objects can be defined as database objects naturally and consistently. Moreover, security checks can be implemented in this layer of objects to protect the data being displayed on the visual objects. We expect the portability of the software will be improved as well.

## System Architecture

Figure 1.

The proposed system architecture of NORA is shown in Figure 1. In order to simplify the discussion, we assume the connection is between two servers. The configuration can be generalized to connections among several servers. For each clinic, there is a Linux server for the database and diald. The Windows 95 clients are connected to the local or remote database server through ODBC drivers. **diald** starts the connection to another server, if needed.

## Configuring a Dial-Up Server

Now we come to connection and configuration. Basically, we will follow the FAQs and man pages on these topics. Since an agent is needed to receive the dial-up call from another computer, we installed **mgetty** to answer the call from the modem. If the call is normal data communication, **login** will be executed to prompt the user on the other side. One of the nice features of mgetty is that it can also act as a fax receiver if the call is a fax, forwarding it to e-mail or printing it. **mgetty** should be started by **init** and specified in inittab.

The user on the other side can start **pppd** after logging in to the Linux server. The options file of pppd should be kept in the simplest form, i.e., IP address, netmask, etc., should not be specified. This is necessary because this configuration is used by any execution instance of pppd, even one started by diald. For example, if an IP address is specified in the options file, the dial-out connection (by diald) IP address will be fixed to the same address. It is incorrect, since this address should be assigned by the target server when the server dials out. A suggested PPP option file, called options, is shown here:

```
proxyarp
lock
crtscts
modem
```

Listing 1.

We leave the other configuration options to the connection script. An example of a script for the dial-in PPP startup, called **startppp**, is shown in Listing 1. It should be noted that **defaultroute** may not be necessary for dial-up from a stand-alone PC, but it is necessary for a LAN connection. Otherwise, there is no way for the dial-up connection to route to all machines in the LAN. Moreover, we skip the connect script for pppd since it can be found in the FAQs and man pages.

You may check the completion of this phase by dialing in to the server with a Windows 95 or another Linux PC using PPP. Start PPP by executing startppp after login. The appearance should be similar to dialing in to an ISP.

## Configuring a Dial-On-Demand Server

As mentioned above, we are supposed to configure the dial-on-demand servers in a symmetric manner, i.e., the connection can be started by any server, and once the connection is established, any machine on either side can access the server on the other side, no matter where the connection started. In this case, **IPtranslation**, which is always mentioned in the diald documentation for connection to the Internet, is not needed. This will simplify the configuration of diald.

We install diald on the server and make it automatically start up after the system boot-up. A sample of the configuration file, diald.conf, is shown in Listing 2.

Listing 2.

The **addroute** script is executed when diald starts. All the accesses to the remote site are routed to the IP address that activates diald (aaa.bbb.fff.eee in diald.conf). For example:

```
#!/bin/sh
/sbin/route add -net aaa.bbb.fff.0 \ # remote LAN
                                                # IP address
netmask 255.255.255.0 gw $4 \
window 2048 metric $5 dev $1
```

We skip the connect script for diald, since it should be similar to the connect script for pppd. It should be noted that the default gateway of all clients in a local LAN should be changed to the IP address of the local server. You may check the completion of this phase by accessing (e.g., using ping, FTP or TELNET) the remote server from any PC in the local LAN. The modem will dial out, and the server will successfully log in to the remote server. However, you will receive no response from the remote server if it has already configured diald in the same way.

### System Integration

The reason for the blocked connection between two servers with diald can be explained in the following way. Since both servers are ready to connect to the other side, the route tables of both servers have a route to the IP address for activating diald if the destination of any packet is to the other side. Therefore, for example, serverA starts the connection to serverB and sends packets to serverB after the connection is established. However, the return packets cannot come back to serverA, since the default route in serverB to serverA is to start diald in serverB. ServerA cannot receive the return packets from serverB through the established connection.

We solve this problem by using the **ip-up.local** and **ip-down** scripts of pppd. **ip-up.local** is executed whenever pppd establishes a connection successfully. So, we delete the route table entry to serverA in serverB when pppd starts a connection. Here is a sample of ip-up.local.

```
/sbin/route del aaa.bbb.fff.0 # remove remote IP
                                  # route
```

It should be noted that the script has no negative effect on the dial-up service from a stand-alone PC. When the connection is finished, the route table should

be recovered. So the ip-down script file, which is executed when pppd stops the connection, is used like this:

```
#!/bin/bash
/usr/lib/diald/addroute sl0 \
255.255.255.0 \
aaa.bbb.ccc.ddd \       # server IP address
aaa.bbb.fff.eee \       # IP address will be activated
                        # by diald
1
```

Basically, it is the same as the addroute script with the same parameters as when diald starts. After this phase, you should be able to access the remote server from any machine in the local LAN. The remote server can access the local server at the same time. Moreover, a stand-alone PC can dial up and access the server from anywhere.

## Conclusion and Future Plans

NORA has been installed and tested on several sites over a three-month period. There has not been a single crash or failure on the server machines. We also set up other services such as file and print servers (Samba), a web server (Apache), etc. Linux is truly a stable, flexible and extensible OS.

We are planning to use ISDN to replace the traditional phone lines to improve the transmission time for large objects like X-ray photos. We are also porting the client software to Linux, so the stability of the whole system can be improved.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue67/3191.tgz.

**Liu Kwong Ip** (kiliu@netvigator.com) obtained a B.Sc. in Information Technology from the City Polytechnic of Hong Kong. He is now working for Northern Oral Science Institute as a system consultant.

Archive Index Issue Table of Contents

Advanced search

# Using Java Servlets with Database Connectivity

**Bruce McDonald**

Issue #67, November 1999

The persistent nature of Java servlets makes them ideal for database/web technology. Mr. McDonald takes a look at using servlets with PostgreSQL and JDBC.

The Common Gateway Interface (CGI) has and continues to be the most commonly used method for creating dynamic and responsive web pages. The main problem with CGI (that stems from the Hyper-Text Transfer Protocol) is that each new client request results in a new instance of the CGI executable being forked by the HTTP daemon. This can lead to considerable resource consumption on web hosting machinery. Many solutions exist to address this problem, most focusing on keeping the executable persistent between client requests. This has the added benefit of holding open costly resources like database and socket connections.

Java is one of the newest kids on the block, but as a C and C++ coder I really like some of the features of this language. Its object model is nice, it is (relatively) portable, and the class libraries available from both Sun and third parties are extensive. Servlets, a rather fanciful name, is the Java answer to the CGI problem. Servlets are Java classes, loaded and kept resident by the HTTP daemon. When the servlet is loaded, it is initialized by a method call; at that point, database connections can be established and held between client requests. In addition to this, there are a number of useful classes which facilitate the more complex server/browser interactions such as cookies. Unfortunately, the Servlet classes from Sun are still in a fluid state and, therefore, code written now may be broken by future releases. This is a fact of programming life and since this is a small application, not much harm is likely.

## Apache and JServ

Since my site has been running the ubiquitous Apache web server, it was an easy choice to stick with this software. The Servlet part was the more tricky

decision. I decided to play it safe and get the Apache-Java solution (http://java.apache.org/). This is not a finished product, but it certainly looked acceptable, and I wanted to have access to the source if anything broke. Alternatives include Live Software's JRun and IBM's Servlet Express. JRun is able to work with a number of web servers including Apache and Netscape, and the code is available under a fairly lenient license. Servlet Express is more of an unknown quantity—that is, I did not even evaluate it since it is only available for Apache 1.2.

The source code for JServ is available for download from the Java-Apache site, and the version I used was 0.9.11. Unpack the code using **tar** into a suitable directory (/usr/local/lib/JServ on my machine) and read the documentation carefully. A Makefile does most of the magic. The Makefile does need to be edited to modify the installation directory—alas, no **autoconf**. A recent JDK is required—I used Steve Byrnes 1.1.6-5 which I have found to be both fast and stable. I also have the **tya** 1.1-3i JIT (Just In Time Compiler) which quite transparently super-charges the execution of Java code, but it is not necessary. Make the JServ code to compile the Java source files to a local directory. Included in the JServ package is the requisite Servlet code from Sun. Both the Sun Servlet code (in the form of a jar file) and the JServ code need to be added to your **CLASSPATH** variable for both development and Apache.

Now it's Apache's turn. Get a nice, recent copy—I have 1.3.1, which I find to be a splendid creation—and unpack the source code into another directory. The magic of building an external Apache module is described in the INSTALL file in the root of the /apache directory. In order to compile the Apache daemon, it is first necessary to “configure” the Apache build environment. This is done through the wonderfully arcane autoconf-produced script **configure**. You merely need to add the following to the ./configure line:

```
--add-module=/usr/local/lib/jserv/mod_jserv.c.
```

This will produce a Makefile that will copy the mod_jserv.c file to the ./src/modules/extra directory and compile it into Apache. Note that merely copying the mod_jserv.c file into the extra directory will *not* accomplish anything. In order to make this new module shared (assuming DSO in apache is configured), add the following line to the configure command line:

```
--enable-shared=jserv
```

This option will produce a .so shared object which is dynamically linked to Apache when necessary.

Obviously some care is required. I cribbed most of the details from the apache.spec file from the apache SRPM; I advise you to get this file and install it.

Make the changes in the spec file and rebuild the RPM (**rpm -ba apache.spec**), then install the new RPM. This applies only to RPM-based systems (Red Hat, SuSE, Caldera, etc).

## Configuring and Testing Apache/JServ

Listing 1.

A number of tags need to be added to your httpd.conf file. This is situated in /etc/httpd/conf in my only slightly modified Red Hat 5.1 machine. Listing 1 is an excerpt from my file. All tags in Listing 1 are covered in the JServ documentation, and the JAVA_COMPILER is really only applicable to systems which have a JIT installed. Notice that all the relevant classpaths are included: there is no need for Swing and other such visual classes to be present. I also set the LD_LIBRARY_PATH to the Java shared objects. If your Java code relies on shared libraries not located in the directories listed in your /etc/ld.so.conf file, then you must add these directories to LD_LIBRARY_PATH. Additionally the JServ error log (ServletErrorLog) is an important configuration item. It is in this file that all JServ-related errors and exceptions are dumped. Keep an eye on the regular httpd ErrorLog file, too. During testing, I run the command **tail -f** on the JServ error log file.

Your current Apache daemon must now be restarted. How you do this will vary depending on your distribution, but for Red Hat systems simply use the SysV scripts to do the magic:

```
/etc/rc.d/init.d/httpd stop
/etc/rc.d/init.d/httpd start
```

In general, it should be possible to merely kill all instances of the httpd daemon and then rerun the new executable:

```
killall httpd
/usr/sbin/httpd
```

Now let's get a little servlet to test the current functionality of the JServ-enabled Apache. The Java servlet in Listing 2 illustrates the benefits and simplicity of the servlet model. This code keeps a counter (hits) that is reset to zero when the servlet is initialized. Every time the servlet is invoked by Apache, the counter is displayed and incremented. Notice that no effort is made to keep the servlet and its data persistent.

Listing 2.

Every servlet must provide a **service** method which gets called whenever Apache needs to serve the URL http://roger.dodger.org/servlet/ServletOne. The

two arguments passed, **HttpServletRequest** and **HttpSerletResponse**, encapsulate the HTTP connection. Information on HTTP-specific data such as cookies can be manipulated with these two objects. The method **init** is called only once when the class is first loaded (or reloaded) by Apache. This initialization method can be used to set up long-lived and potentially costly resources. From this simple example, it can be seen that the class, once loaded by Apache, remains persistent until it is reloaded. The *hits* variable is initialized once (in the init method), and is then incremented each time service is called. JServ will reload the class if it has been modified since it was first loaded. Sometimes it is necessary to reload the class if a change is made to the property file; in this case, **touch** the class file. This happens quite transparently.

### PostgreSQL and JDBC

Servlets are most often mentioned in the same context as database connectivity, and rightly so. Servlets and their persistent nature make for ideal database/web integration technology. My first forays into this led me quickly to the conclusion that CGI scripts were too slow for anything but large-scale processing with databases—where the initial setup of the database connection matched (or even exceeded) the actual database operations. Since I was familiar with PostgreSQL which came with my Red Hat 5.1 installation, I decided to use that RDBMS for experimentation. I also had some experience with developing Java database client software. The Java Database Connectivity (JDBC) classes in the JDK (Java Development Kit) are easy to use and somewhat portable between databases. The JDBC classes are comprised of database-independent code (the java.sql.* classes, part of the JDK) and database-specific code. The JDBC classes for PostgreSQL are part of the later distributions and are necessary for this application.

Listing 3.

All this can be clearly demonstrated by a small JDBC example. In order for this to be a useful example, it is necessary to have a working PostgreSQL installation. Listing 3 is a rather dense example that illustrates a number of JDBC issues. First, the setup of the PostgreSQL JDBC driver (**loginJdbc**) and the login to the database (**loginUrl**, **loginUser** and **loginPasswd**). The variable **loginUrl** is often tricky to get right (especially when learning). The last part, in this case “dbname”, is the name of the database to which the connection is being made. The latter two constants (**loginUser** and **loginPasswd**) need to be changed to reflect your database environment. Setting the permissions to the database involves executing the command **createuser** (as user postgres) and then *granting* the needed permissions to the user. Once the connection is established, the SQL SELECT statement is executed and the result set is captured by the variable *rs*. Meta-data for this result set (*rsmd*) is then used to

determine the number of columns returned and to display the column names. The result set is then iterated through, printing each row as strings. This example is very simple-minded and is intended only for SQL SELECT statements because of the result set. Be prepared for exceptions; e.g., an exception is thrown if no columns are returned.

### The Guest Book Application

My first guest-book application was a Perl CGI script that kept all the guest entries in a formatted text file that was displayed or added to as the parameters dictated—simple and effective. Unfortunately, I wanted to do more with that information, including linking back to the Apache log files and maybe even handing out cookies to monitor usage. The script, which started small and lean, began to choke on the file processing necessary. It was also a little distressing, since I had a very functional RDBMS installed and ready to use. I studied the alternatives (**mod_perl**) and decided to go with Servlets. The first step was to design the tables. The most important table holds each guest-book entry on a row.

```
CREATE TABLE guest_book (
        entryid         INT NOT NULL,
        response        CHAR(8),
        name            VARCHAR(32),
        addr            VARCHAR(48),
        email           VARCHAR(24),
        time            DATETIME,
        comment         TEXT,
        PRIMARY KEY (entryid)
)
```

Each row in the guest-book table is uniquely identified by the **entryid** column which has an implicit index. Additionally, the handling of these keys is handled by a key table, a convenience in a small application but almost essential in larger applications.

```
CREATE TABLE key_table (
        id                      INT NOT NULL,
        val             INT DEFAULT 0,
        PRIMARY KEY (id)
)
```

The key table is able to keep track of any number of keys within the range of the **int** type, which in my case is [-231,231-1]. In order to get a new key, the current key must be retrieved and then incremented. This must be atomic, i.e., the operation must take place in a transaction. This is illustrated in the **getKey** method. Ideally, a stored procedure would handle all these details (the key-table concept should not be visible to application developers in an ideal world), but this level of detail with PostgreSQL involves C shared libraries and Database APIs—something with which I did not want to get involved.

An additional problem that needed solving was how to provide servlets with property files in a consistent fashion. Property files provide a convenient way of placing data that would otherwise be hard-coded into the application. This is done by providing a property on the command line to JServ called base.dir which points to a world-writable directory (but is **chmod +t** so that user's may not stomp on other users files). This is specified in the httpd.conf file in the **ServletBinaryArgument** tag. In this directory you can store property files which may be loaded by servlets. I am sure this can be done in a smarter fashion.

The main processing entry point is the service method. From here, either the list of entries is displayed (**listEntries**), a form is displayed (**showForm**) or a new entry is made (**addEntry**). The code for the Guest Book application is not shown due to space considerations, but is included in the archive file at ftp.linuxjournal.com/pub/lj/listings/issue67/3243.tgz.

Listing 4.

Listing 4 is the property file used to set up the various parts of the servlet. I moved a number of the HTML header and footer strings into this file along with the JDBC configuration parameters.

## Conclusion

The servlet/JDBC/PostgreSQL proved to be a powerful and fast technology. Most of the problems I encountered were configuration problems that required me to carefully read the associated documentation. Unfortunately, technical documentation on Servlets is scarce and I would urge further experimentation. The next step I took with this application was to install Sybase ASE for Linux, a RDBMS with which I am comfortable. I then coded a number of stored procedures that allowed the Servlet to delegate most of its data manipulation to the database, where it rightly belongs. If you are going to replace PostgreSQL with Sybase, it is necessary to get the jConnect JDBC classes from the Sybase web site. Of course, this can be done with PostgreSQL, but learning the details of an RDBMS C API was tangential to the exercise.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue67/3243.tgz.

**Bruce McDonald** (bruce@triphop.dyn.ml.org)

Archive Index Issue Table of Contents

Advanced search

# Open Database Connectivity

**Peter Harvey**

Issue #67, November 1999

Mr. Harvey describes the ODBC open specification for application developers.

Open Database Connectivity (ODBC) is an open specification for providing application developers with a predictable application programmers interface (API) with which to access data sources. Data sources can be just about anything, provided someone has created an ODBC driver for it. The most common data source is an SQL server.

The two major advantages of coding an application with the ODBC API are portable data access code and dynamic data binding.

## Portable Data Access Code

The ODBC API (or CLI, command-line interface), as outlined by X/Open and the ISO, is available on all major platforms. Microsoft platforms include many enhancements to this specification. The current version from Microsoft is 3.51. The idea is that a programmer using the ODBC API is likely to have data access code which is portable to other platforms. The same code will also be portable across different data sources. For example, data for an accounting program application can reside on a light SQL server during development and then be moved over to a heavy SQL server just by linking to a different ODBC driver. ODBC delivers platform and data source portability.

## Dynamic Binding

Dynamic binding allows the user or the system administrator to easily configure an application to use any ODBC-compliant data source. This is the single biggest advantage of coding an application with the ODBC API and purchasing such an application. Dynamic binding allows the end user to pick a data source, e.g., an SQL server, and use it for all data applications. Applications do not have to be recompiled or recoded for the new target data source. This is

achieved by the ODBC Driver Manager which will pass the ODBC calls to the user's ODBC driver without the need to relink the code. ODBC enables the user to choose where the data will be stored.

### unixODBC Project

The unixODBC Project's goals are to develop and promote unixODBC as the definitive standard for ODBC on the Linux platform. This is to include Microsoft extensions, where they make sense, and GUI clients. The unixODBC team is achieving this objective by providing the best technical solution to ODBC demands on the Linux platform. All unixODBC development is released under GPL or LGPL.

The components of this project are the Driver Manager, DataManager, ODBCConfig, Odbcinst, drivers and other utilities.

### Driver Manager

Figure 1. Libraries

This share library is the hub of most ODBC activity, but its function is simple. Ninety percent of the Driver Manager's function is to validate arguments, load and unload drivers and pass the call to the driver in a manner consistent with the ODBC specification. Normally, an application links only to this share to get the ODBC support it requires (see Figure 1). The Driver Manager loads/unloads the appropriate driver and passes calls to the driver.

### DataManager

Figure 2. DataManager TreeView

This is a GUI-client utility. The current version is based upon Troll Tech's Qt class library (http://www.troll.no/). The DataManager allows the user to browse and manage data sources (see Figure 2). The right side of the TreeView contains a sizable canvas which can be extended to include properties for any TreeView selections. An example of this has been implemented for the data source TreeViewItem. When a data source is selected, the canvas becomes a handy editor which can be used to submit SQL, review results and save/load either SQL or the results. Table designers and data editors could be easily added to the DataManager using the same techniques. The DataManager is an easy way to manage ODBC data-source resources.

## ODBCConfig

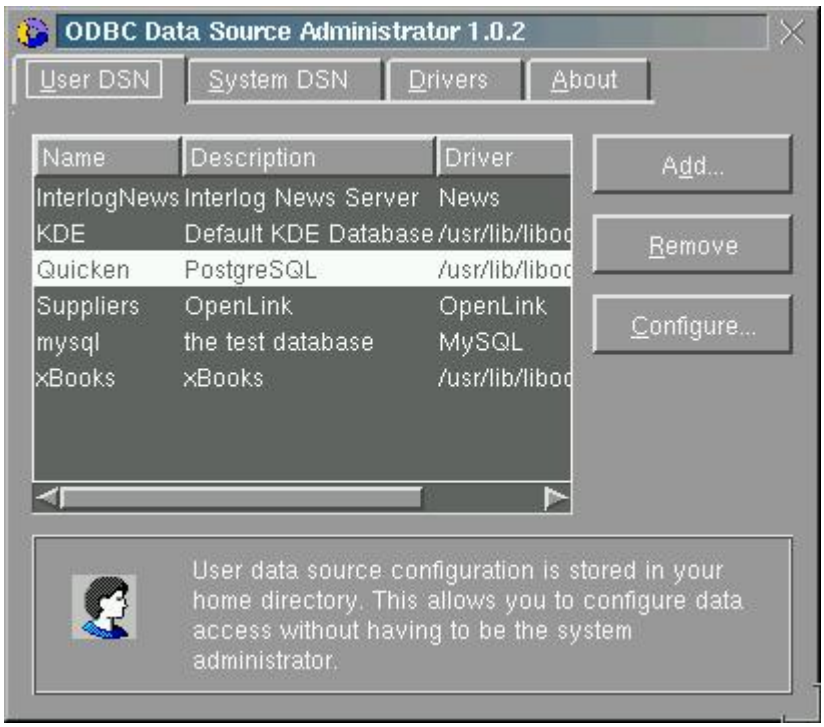



Figure 3. ODBC Data Source Administrator

This is another GUI-client utility. It has been created to be user compatible with the Microsoft ODBC administration utility (see Figure 3). ODBCConfig makes it easy, even for non-techies, to configure their data sources. ODBCConfig uses the Odbcinst library to read/write ODBC system information. ODBCConfig will make use of any installed driver configuration libraries to present a list of driver-specific options to edit. ODBCConfig functionality is an excellent candidate for the KDE (http://www.kde.org/) Control Center. ODBCConfig makes it easy to configure ODBC data sources.

## Odbcinst

Figure 4. ODBC Data Flow

This is a share library which implements many useful Microsoft extensions and a number of unixODBC extensions. It provides an API for reading and writing ODBC system information (see Figure 4). This essentially means reading and writing the various INI files and environment variables. unixODBC also provides a command-line interface to this library. This command-line tool (of the same name) can be used by driver programmers when creating their install script/ RPM. The Odbcinst library is used heavily by ODBCConfig, the Driver Manager and drivers but is seldom used by other applications. Odbcinst implements many Microsoft extensions.

### Drivers

The drivers, and their data source-specific client libraries, typically implement the vast majority of the ODBC functionality. As important as the Driver Manager is, it is the driver that carries out most of the work. The unixODBC Driver Manager has been designed to be binary compatible with any compliant drivers compiled on Linux. unixODBC includes several drivers (and their sources) as examples. unixODBC also includes a driver template for those wishing to write a new driver. MiniSQL, MySQL and PostgreSQL are examples of drivers included in unixODBC. unixODBC maintains a driver-certification process which can be found at its web site (see Resources).

### Other Utilities

unixODBC also includes a library for working with INI files, a LOG library and a command-line tool for executing SQL commands. The INI library is used by Odbcinst for reading/writing the ODBC system information. The LOG library is used by the Driver Manager and Odbcinst to log ODBC activity, such as errors, and implement ODBC tracing. The **isql** command-line tool allows the user to submit SQL commands in a terminal window. isql can also be used in batch mode, which can be quite useful because it can return results wrapped in an HTML table or delimited with a chosen character. It supports piping and redirection.

### How Do I Get Started?

You can download the source distribution from the unixODBC website. Unpack the file using **tar**, then follow the instructions in the README file. You will want the Qt dev libraries to build the GUI components and you will need the database-specific client libraries for any drivers you decide to build. The README talks about this in more detail. You must have at least one driver section in /etc/odbcinst.ini. You can add this section using the Odbcinst command-line tool or your favourite editor. Normally, odbcinst.ini will get updated by driver install scripts using the Odbcinst command-line tool, but at this time, you may have to edit this file directly. A driver section in odbcinst.ini should look something like this:

```
[MiniSQL 2.x]
Description = MiniSQL ODBC Driver
Driver = /usr/lib/libodbcmini.so.1.0.0
Setup = /usr/lib/libodbcminiS.so.1.0.0
FileUsage = 2
```

Now run ODBCConfig to add, edit or remove data sources. You will need to be root to work with system data sources, but any user can add, edit and remove user data sources. Now run the DataManager, and you should see all ODBC data sources. When you try to expand the TreeView below a data source, you

will be prompted for login information to connect to the data source. You can also try to connect to your data source using the isql command-line tool. Simply execute the command:

```
$isql DataSourceName MyID MyPWD
```

### An Application's Perspective

Most database access can be accomplished using a simple handful of ODBC function calls. In fact, it is good practice to keep it simple, because each driver implements its own level of compliance and completeness. An application can expect to be used with very modest drivers from time to time. The isql command-line tool is an example of an application that uses only a small, simple set of ODBC functions. I will not get too deep into ODBC APIs, since you can pick up excellent reference material elsewhere. The typical sequence of events goes something like this:

1. Connect to a data source.
2. Create and Execute an SQL Statement.
3. Process results.
4. Close connection.

### Connect to Data Source

Initialize ODBC by calling SQLAllocEnv and SQLAllocConnect, then call SQLConnect.

### Create and Execute an SQL Statement

Initialize a statement by calling SQLAllocStmt. Call SQLPrepare to allow the Driver Manager a chance to preprocess the SQL, then call SQLExecute.

### Process Results

The simplest way to process results is to call SQLFetch in a loop and SQLGetData for each column in the result set. SQLNumResultCols can be used to find out how many columns are in the result set.

## Close Connection

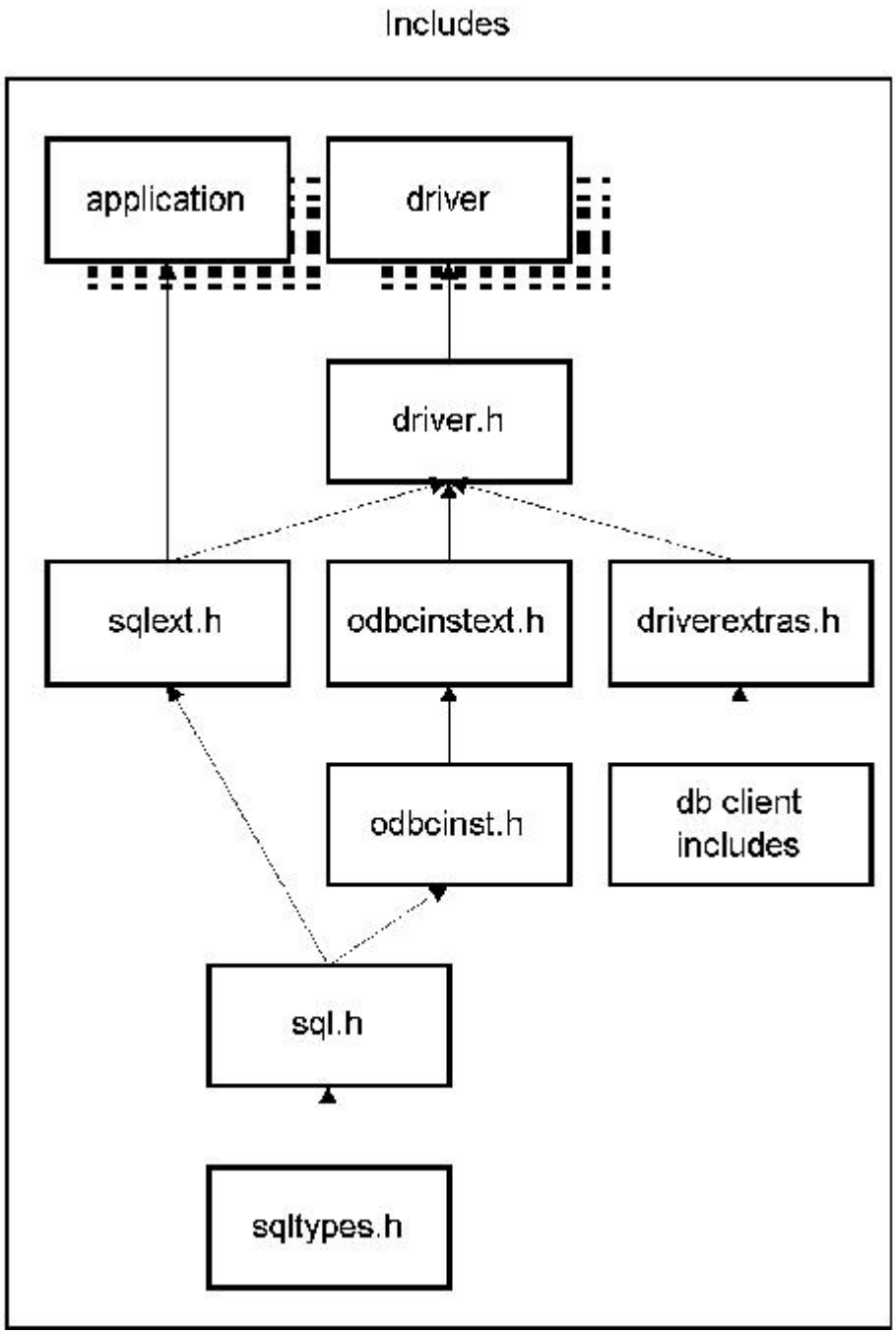



Figure 5. Include Files

Call SQLFreeStmt, SQLDisconnect, SQLFreeConnect and SQLFreeEnv to clean up. Your source should include sqlext.h (see Figure 5) and should link against libodbc.so (see Figure 1).

## A Driver's Perspective

The driver and the Driver Manager share 98 percent of their function names. This is not surprising when you consider that the Driver Manager passes off most processing to the driver. An important difference, however, is that the **Environment**, **Connection** and **Statement** handles contain different information.

For example, the driver's **Connection** handle will often contain some database-specific data such as a socket handle, while the Driver Manager's **Connection** handle will usually contain very little aside from a pointer to the driver's **Connection** handle. unixODBC contains a driver template which should act as a good starting point for anyone interested in creating a new driver. It should be very easy to port a driver to/from Linux using unixODBC, because unixODBC is designed to support driver code from other platforms. A useful driver will implement, at a minimum, the following functions: SQLAllocConnect, SQLFreeConnect, SQLAllocStmt, SQLFreeStmt, SQLConnect, SQLDisconnect, SQLPrepare, SQLExecute, SQLFetch, SQLGetData, SQLNumResultCols and SQLColAttribute.

Perhaps the best way to learn how a driver works is to look at some driver code. unixODBC contains a number of such examples. Your source will likely implement driver.h and driverextras.h, but at a minimum it should include sqlext.h and odbcinst.h (see Figure 5). Driver code from other platforms may have their own version of driver.h and driverextras.h or may not have them at all; this is okay, because these two includes are implemented in the local source directory and are not used by any other code. You should link against libodbcinst.so and any database-specific libraries (see Figure 1). unixODBC has something for just about every Linux user.

Application developers can now create portable data access code independent of the platform and the data source. Driver programmers can use the unixODBC driver template to get started on a new driver, then use Odbcinst when creating their install script/RPM. All Linux users can easily configure ODBC data sources for their applications with ODBCConfig, then take a look at what resources are available in a data source by browsing in the DataManager. Ease of use and functionality surpass other platforms which have had ODBC for a number of years. Most importantly, widespread adoption of unixODBC in popular Linux distributions will allow application developers to assume a desktop has these features and take advantage of them. Such features are of critical importance in having Linux fully accepted on an average user's desk.

Tips

Resources

**Peter Harvey** started programming games on the C64 while in the Canadian military and went on to make programming his occupation of choice. He is the founder of CodeByDesign, which recently changed its focus from developing for MS platforms to developing for Linux. Peter can be reached at pharvey@codebydesign.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL



# MySQL Introduction

**David Axmark**

**Michael xxxxxxx Widenius**

Issue #67, November 1999

A look at the MySQL database—where it's been, where it is now, and where it's going.


MySQL's history goes back to 1979 when TcX, the company that developed MySQL, started working with database programs. This first version was a screen builder/reporting tool written in BASIC. At that time, state-of-the-art computers had 4MHz Z80 processors and 16KB of RAM. This tool was moved to UNIX and further developed during the following years. In the mid-1990s, we started having problems with customers who liked the results the tool produced but wanted something they had heard about before (a buzzword was needed). So we started looking at making an SQL (an appropriate buzzword) front end to our low-level libraries. We found mSQL, but it did not work for our purposes. So we started to write an SQL engine from scratch. However, since the mSQL API was useful, we used it as the basis for our own API. This made it easy to port some applications we needed that were available for the mSQL API.

Since this tool would be usable by others, we decided to release it according to the business model pioneered by Peter Deutsch at Aladdin Enterprises with Ghostscript. This copyright is much more free than the mSQL copyright and allows commercial use as long as you don't distribute the MySQL server commercially.

It is not perfectly clear where the name MySQL came from. We have used the prefix “my” for libraries and path names since the mid-1980s. The main MySQL

developer's daughter is named My—a fairly common name among Swedish-speaking Finns—so naming our database MySQL was very natural.

In May 1996, MySQL version 1.0 was released to a limited group of four people, and in October 1996, MySQL 3.11.1 was released to the public as a binary distribution for Solaris. A month later, a Linux binary and the source distribution were released. The MySQL release included an ODBC driver in source form. This also included many free MySQL clients ported to MySQL.

### Platforms

The initial version of MySQL worked only on Linux and Solaris. The biggest problem in porting to other platforms was that MySQL needed a working POSIX thread library; in January 1997 a modified version of MIT-pthreads was included in the distribution.

Table 1.

### MySQL Language APIs

Table 2.

To be able to use MySQL from your favorite language, you need an API. The first MySQL version included only C and Perl APIs. Now there are many (see Table 2). With the exception of the Java API, all of these use the C API to communicate with the MySQL server. So, as you can see, MySQL can be used from many popular languages.

### MySQL Benchmarks and crash-me

When we had gotten a nice working system, we wanted to test it against old versions and against other databases, so we started looking for good benchmarks. We found that most benchmarks (like the TCP ones) represent an SQL server's performance as a single number, often as transactions/second. We regard these to be almost worthless, since comparatively few users run applications that do the same thing as these benchmarks. There is usually no way to determine your application's performance from the numbers given by this type benchmark.

The MySQL benchmarks are designed to show how fast a SQL server is for common operations, such as establishing a connection, performing simple inserts or joining two tables using a key. This also makes it possible to calculate loads on a web site when you know the mix of operations. Of course, you need to actually understand your own application to judge its performance with any database.

Over time, we got many requests on the MySQL mailing lists about MySQL's features and how it compares feature-wise with other databases. As Michael (the main developer) didn't want to dig into old inaccurate reference manuals to find this out, he thought of a program that automatically detects what a SQL server has to offer. He also thought it would be a nice test of how stable the MySQL code is when you start to send it “abnormal” queries.

While working with the benchmarks, we needed a list of capabilities for all supported databases. Since doing this by hand was *very* tedious, we made a tool to do it automatically. While trying early versions of this tool on some different servers, bad things happened—the servers crashed. All this crashing led us to name this tool **crash-me**. In fact, the only SQL server that has gone through this testing without crashing is Oracle. Of course, all bugs found in MySQL were fixed immediately.

**crash-me** (and the benchmarks) are implemented as a Perl DBI/DBD program that sends thousands of queries to a database to find out how things work in real life. By doing this, it finds many limits in the server, such as the supported column/query/variable/index lengths.

**crash-me** is also a nice tool for helping you write portable SQL, since it can provide a list of which functions, types and limits exist in the server you wish to use. Currently, we have crash-me results from Adabas-D, Access, DB2, Empress, Informix, MS-SQL, MySQL, Oracle, PostgreSQL, Solid and Sybase. As the crash-me table is big and very detailed, we will not include it here, but it is available at http://www.mysql.com/crash-me-choose.htmy/.

What operations does the benchmark test? First and foremost, the basic SQL operations are tested: **INSERT**, **UPDATE**, **DELETE** and **SELECT**. Other tests include a connect followed by a select, and creation of tables and indexes.

The individual tests should give a good indication of how fast an SQL server is for that specific operation. Do not use the “total time” as an overall measure of the value of an SQL server. This is because the tests are not weighted against each other. Some tests are run more times with different options and numbers of rows. An SQL server may be extremely bad at some “unimportant things”, while it's very good at the things for which you actually intend to use it.

We use the total time to compare things like the same database engine on different operating systems. We also use it to see how new versions of MySQL stack up against old ones.

Since all benchmark tables take even more space than the crash-me results, we include only a few observations on how well MySQL runs on different platforms.

- Linux 2.2 is much faster than Linux 2.0 on a multi-CPU machine. This is because the Perl and the MySQL server run on different processors and the new SMP code is faster.
- Linux is 7% faster than Windows 98 and 49% faster than NT on the same machine.
- Windows 98 is 27% faster than NT on the same machine.
- A Pentium II 400MHz machine running Linux 2.2 is much faster than a Sun Ultrasparc 2/CPU 400MHz machine running Solaris 2.7. The primary reason for this difference is that Linux caches the file system much better than Solaris; this result might be different under higher load. We will include a threaded test in the next generation of benchmarks to test things like this.
- If you do many inserts on Solaris, you will get only a 22% speed increase by upgrading your processor speed by 100%.
- The overhead of using MyODBC, and probably any ODBC driver, is at least 19%.

Note that while benchmarking, it was still possible to work on the Linux machine without any problems. However, NT became so slow that it was impossible to do any other work, even simple editing. It took about 30 times longer to start up a new DOS window, and we had to wait 10 seconds or so before typed characters showed up.

There are still many things to be done for both crash-me and the benchmarks. For example, we would like crash-me to report if there are identical functions that do the same thing (such as, instead of **CONCAT** one can use "||"). Also, many new tests should be added to test which sub-select constructs an SQL server allows. Of course, the documentation and presentation of the results could be much improved.

Both these tools give invaluable information to any developer who uses more than one SQL server. If they do not test the feature you need, please contribute a new test. More test results can be found at www.mysql.com/benchmark.html.

### Why MySQL Is Fast

If you checked out the benchmark page above, you will notice that MySQL is many times faster than the competition on almost everything. Why is this so? Some of the speed comes from things that are not supported in MySQL, such as transactions, foreign keys and triggers.

Because the MySQL server is coded mainly by one person with many years of coding experience, very little redundant code is in it. Most of the basic algorithms also come from an era of slow CPUs and small amounts of memory. The algorithms have mostly been extended to use larger caches if there is available memory.

As a result, MySQL has a compact fast design (the code size of the server is less than 1MB on an i386) which normally uses very little memory, but can be configured to take advantage of large amounts of memory.

MySQL has many useful optimizations for maximum speed. Some examples are:

- Most joins are done in one sweep.
- Very few normal joins require temporary tables. Joins involving a **GROUP BY** and an **ORDER BY** on something other than the **GROUP BY** part will create a (usually memory-based) temporary table.
- Memory-based temporary tables have dynamic hashing.
- MySQL has a user configurable key cache and a record cache to quickly scan tables. Open tables are cached in a table cache.
- An index optimizer quickly finds which possible index to use with a specific **WHERE** clause.
- A range optimizer finds the range for an index which will then be used to find the matching rows. The range optimizer can optimize queries that use a combination of **>=**, **>**, **=**, **<=**, **<** and **LIKE** (of type “keyword%”) on a key column. When many possible indexes can be used, the range optimizer will choose the index that matches the smallest number of rows.
- A join optimizer rearranges the tables in a **SELECT** statement in the optimal order. In the rare case the MySQL join optimizer gets this wrong, one can force the optimizer to join the tables in a particular order with the **STRAIGHT_JOIN** keyword.
- For each sub-join, a simple **WHERE** is constructed to get a fast **WHERE** evaluation for each sub-join and to skip records as soon as possible.
- A **WHERE** optimizer removes constant conditions.
- Early detection finds invalid constant expressions. MySQL quickly detects that some **SELECT** statements are impossible and returns no rows.
- SQL functions are implemented through a highly optimized class library. Almost all parsing and calculating is done in a local memory store. No memory overhead is needed for small items, and the normal slow memory allocation and freeing is avoided. Memory is allocated only for unexpectedly large strings by using **malloc** and **free**.
- In queries of type

```
SELECT ... WHERE col IN (a,b,c,d,e,f,...)
```

- the **IN** part is sorted and checked through binary searching.
- **MIN/MAX** on an indexed column is done with one key fetch.The MIN/MAX optimizer can also find the best value when one has specified all preceding index columns in the **WHERE** condition:

```
SELECT MAX(index_part2) FROM tbl_name WHERE
index_part_1=constant;
```

- **COUNT(*)** on a single table without a **WHERE** is retrieved directly from the table information.
- If all used columns for some table are numeric and form a leftmost prefix for some key, the values may be retrieved from the index tree rather than the data file for greater speed.
- Indexes are used to find rows matching

```
WHERE index_column LIKE "prefix%".
```

- **GROUP BY** and **DISTINCT** are optimized by creating a temporary **HEAP** table with the **GROUP BY** (or whole row) as a unique index.
- **INSERT DELAYED** inserts rows in a queue to the SQL server, which is useful if you are using MySQL for logging and can't afford to wait until the INSERT is completed.
- Index compression is used to get smaller and faster indexes.

### No Transactions and Many Simultaneous Users

One of the design decisions that has resulted in the greatest number of questions is the lack of transactions. There are, of course, applications that require transactions to work, but a wide range of applications work very well without them.

Some people believe that since MySQL does not support transactions, it cannot support many simultaneous users. Each MySQL client gets a dedicated thread in the MySQL server, which allows different users to access the same tables at the same time. All MySQL operations are atomic: no other users can change the result for a running query.

When designing MySQL, we had a greater need for speed than for transactions. It's no use having transactions if the SQL server becomes so slow it's unusable for what you need to get done.

Another common misunderstanding is that transactions provide extra robustness through the redo logs. This extra security can be had by simple means in MySQL. That is, do normal backups and then apply the update log to the backup. The update log is a file containing all SQL statements that change any data.

MySQL also has external utility programs to check, optimize and repair individual tables.

## MySQL User Base

As MySQL has about 50 mirrors over the world, and we don't get download statistics from them, it's hard to tell how many MySQL installations are out there.

The WWW and FTP log at http://www.mysql.com/ gives us the information shown in Tables 3 and 4; all counts are based on the number of distinct IPs.

Table 3.

Table 4.

In the Linux community, many sites use MySQL as a back end for dynamic web pages. Among those are http://slashdot.org/, http://freshmeat.net/ and http://www.linux.com/.

On linux.com, every page does somewhere between 10 and 20 queries to the database. And linux.com does anywhere between 500K and 800K page views per day. They run MySQL on its own server, a dual Xeon system with huge amounts of RAM and hard-disk space.

While writing this, I asked *Linux Journal* what they use as a web back end, and learned they also use MySQL. Among the awards we have been given, we highly value the “Most Used Database” 1998 award we got from *Linux Journal*'s readers.

## Where MySQL Is Today

- Client/server
- Multi-threaded, multi-user and very fast
- APIs to many different languages
- A good, free ODBC driver
- Very portable
- Many different column types which support all ANSI 92 and all ODBC 2.50 types as well as some new ones
- Support for almost all ODBC 3.0 and SQL ANSI92 functions
- Full support for SQL **GROUP BY** and **ORDER BY** clauses; support for group functions (COUNT, AVG, STD, SUM, MAX and MIN)
- Ability to mix tables from different databases in the same query

- Very flexible privilege system where privilege is based on host and user
- Support for **LEFT OUTER JOIN** with both ANSI SQL and ODBC syntax
- Fixed-length and variable-length records
- Handles large databases; at TcX, we are using MySQL with some databases that contain over 50 million records.
- Very robust with no memory leaks; all reported memory leaks have been in non-MySQL libraries, most notably some versions of glibc.
- Ability to configure many different character sets, e.g., Japanese/Chinese
- Error messages available in many languages
- Many utilities and much contributed software
- MySQL is extensively documented. Most questions can be resolved by reading the MySQL manual. We try to document *everything* to avoid getting too many questions on the MySQL mailing lists. The current manual has recently been improved considerably, thanks to the great work done by Paul DuBois.
- Many small, extremely useful extensions that help you get your work done

### New in the Latest Development Version (3.23)

- Binary portable table format—it is now possible to copy MySQL table files between different architectures.
- More and longer indexes—maximum is 32 which can be 500 bytes long (16/128 previously).
- Even better index compression—it is faster and uses even less disk space.
- Indexes on **BLOB/TEXT** columns just like a **CHAR** column.
- Support for tables greater than 4GB on file systems which support files that big. The new limit is about 9 million terrabytes.
- Has better fragmentation handling for the dynamic row format.
- Added in-memory tables with hashed keys—an extremely fast way to have lookup tables.
- Allows true floating-point columns with values such as 1.0E+10.
- Includes example C code for a procedure that analyses the result from a SELECT.
- Faster **SELECT DISTINCT** handling has been added.
- Added much useful information in **SHOW TABLE STATUS**.
- **CREATE TABLE (...) SELECT * from *a,c* where *something***. This creates a table using data from a **SELECT** in one step. The data types and field names are automatically generated from the **SELECT**.

- Removed the old limitation with big **GROUP BY** queries (with **SQL_BIG_TABLES=0**) that resulted in a “table is full” error.
- Loads BLOBS from files with the **LOAD_FILE** function.
- COUNT(DISTINCT) is supported.

### Future

We have no intention of stopping development of MySQL. Over time, MySQL will be 100% ANSI 92 compatible. As we still want MySQL to be fast, we will always give the user the option of specifying the removal of features which make a normal SQL server slow.

As an example, the GRANT system will not have any speed impact unless you use this to restrict table or column access.

The current “TODO” list can be found in the MySQL manual at www.mysql.com/doc.html. Everything in this list is in the order we plan to implement it.

### The MySQL License

We have worked many years with GNU/BSD and other programs from the Net and have always believed that programs should be available in source. Because of this, we chose to use the same license as Aladdin GhostScript for the MySQL server on UNIX, and we made the client's completely free.

By the time this article reaches publication, there should be an old version of MySQL (3.20) with a GPL copyright available. We will continue releasing old versions under the GPL.

This means that for normal (even commercial) internal use on UNIX systems, MySQL costs nothing. You do not have to pay us if you do not want to. A license is required only if:

1. You sell the MySQL server directly or as a part of another product or service.
2. You charge for installing and maintaining a MySQL server at some client's site.
3. You include MySQL in a distribution that is non-redistributable, and you charge for some part of that distribution.
4. You use MySQL on a Win32 (Windows 95, 98, NT or Windows 2000) system.

For circumstances under which a MySQL license is required, you need a license per machine that runs the **mysqld** server. However, a multiple-CPU machine needs only one license, and there is no restriction on the number of MySQL

servers that can run on one machine, or on the number of clients concurrently connected to a server running on that machine.

The following points set forth the philosophy behind our licensing policy:

- The SQL client library should be totally free so that it can be included in commercial products without limitation.
- People who want free access to the software into which we have put much work can have it, as long as they do not try to make money directly by distributing it for profit.
- People who want the right to keep their own software proprietary, but also want the value from our work, can pay for the privilege.
- That means normal in-house use is *free*. But if you use MySQL for something important to you, you may want to help further its development by purchasing a support contract or by contributing documentation, code samples or something else.
- Our policy is that no one should have to pay for normal upgrades. In the future, we may require a new license for major upgrades with major new features (like transaction support). This means that in the long run, MySQL will be a very good investment compared to other databases.

As Win32 is a highly commercial OS with very high development costs (and development pains), we see no other alternative than to provide MySQL-Win32 only to paying customers, users who have helped us with MySQL in some way or users who think they can contribute to any part of MySQL. If we did this in any other way, we could not afford to continue developing MySQL on Win32 or even keep this version up to date with the UNIX version. In effect, we let users who run Win32 pay for the development of tools of our other operating systems.

**David Axmark** (david@detron.se) lives in Uppsala, Sweden with his plants and computers. He has been working as a software consultant for over 15 years.

**Michael Widenius** lives in Helsingfors, Finland with his wife and his two kids My and Max. He also has been working as a software consultant for over 15 years.

Among the things both authors have worked with are software for a one-card computer used by power companies, a video-rental system, a state-of-the-art market research system, advanced business graphics, a word processor that could handle Z80 Assembler+Basic, and a full operating system for an 8-bit computer (Z80) and many other other projects.

Archive Index Issue Table of Contents

Advanced search

# Oracle Database Administration with Orac

**Kevin Kitts**

Issue #67, November 1999

Database Administration got you down? Here's a tool to help you out.

Orac is an open-source database administration tool written in Perl/Tk. It was written primarily by and for DBAs (database administrators). However, it will also be very useful to developers and anyone else who wants to understand more about how (and how well) their database is working.

Orac provides much of the functionality that any DBA could want. It includes scripts that help in managing physical database files, users, database objects (such as tables, views, sequences, etc.) and scripts that help tune the database and resolve “locking” conflicts.

Orac builds on this collection of widely available SQL scripts by providing a nice GUI and a logical organization of the scripts. Orac is extensible, of course, in the sense that the source code is readily available. Even better, though, Orac has an easy method of adding support for more SQL scripts without editing even a single line of code. So for those DBAs out there who already have their favorite scripts, Orac is even more useful.

By the way, the Orac program was named after a supercomputer in the BBC science fiction television series “Blake's 7” and in no way takes its name from the Oracle database or the Oracle Corporation. No affiliation with Oracle Corporation is intended or implied.

## Why Use Orac

Many experienced DBAs manage databases almost exclusively with command-line utilities. Oracle provides a number of tools such as **sqlplus** for querying the database, **svrmgrl** for startup and shutdown of the database and **sqlldr** for loading ASCII files. Like the Oracle database, these simple command-line tools have proven themselves solid, reliable and efficient.

Unfortunately, the same cannot always be said for the various high-end database administration tools on the market. Configuration and setup can be difficult. They are often slow, and at times it may not be clear exactly what the tool is doing. Some commercial DBA tools even require the use of a proprietary scripting language. (By the way, I'm running Oracle 8.0.5, WordPerfect 8 and Orac 1.1.11 on a Pentium 75/64M, and it works just fine for single-user experimentation.)

Orac provides an elegant way to capture and organize the various scripts many DBAs need to do their job. It represents a middle ground between having a bunch of ad-hoc scripts executed at the command line and the complicated commercial tools.

There will always be a role for the commercial offerings, of course, and you might eventually decide to purchase one. However, the knowledge gained from having used a free tool like Orac in a real-world setting can only be a benefit.

## How Orac Works

Orac is a Perl script that performs two basic tasks. It retrieves information from the database and presents the information to the user. A couple of important Perl modules are used to do this work. First, DBI.pm is used to make the connection to the database. Here is a simple code snippet that connects to the Oracle database called ORA1 and gets a list of files that make up the database.

```
#!/usr/bin/perl -w
use DBI;
$dbh = DBI->connect('ORA1','ADMIN','ADMINPASS','Oracle');
$sth = $dbh->prepare
   ("select file_name from dba_data_files");
$sth->execute;
while (@row = $sth->fetchrow()) {
   print "File Name: @row\n";
}
$sth->finish;
$dbh->disconnect;
exit;
```

Of course, Orac reads its SQL from a file instead of hard-coding the statement into the Perl script, but the basic principles remain the same.

Listing 1.

Once the needed data are in hand, it is fairly straightforward to display them using the routines in the Tk module. The script in Listing 1 is similar to the one above, but instead of using a simple **print** to send the information to standard output, it uses Tk to display the results in the X Window System. Figure 1 shows the results from running the Perl/Tk script.

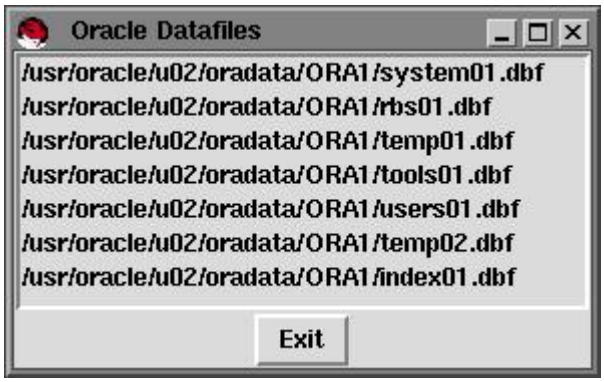



Figure 1. Output Display of Listing 1

Again, Orac uses Tk in a very flexible way. Orac loads its menus from a text file after the program starts. To recap, Orac loads both the SQL scripts it executes and the menus making up the program from text files, after the program starts. Any idea where this might lead? More on this later. Figure 2 shows how SQL gets executed in Orac.

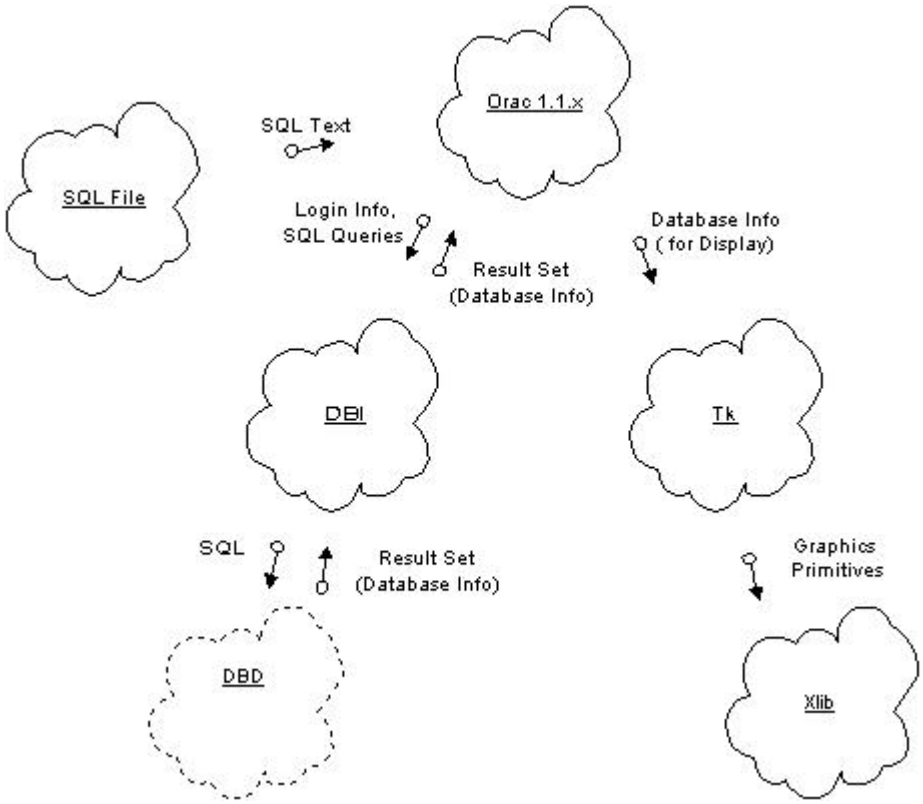



Figure 2. How SQL is Executed by Orac

## Orac in Action

A number of common tasks are faced by DBAs, such as the management of users, database performance, and of course, the actual database files. We'll take a look at the last item, database file management, to show how Orac can be used to make this task easier.

A full explanation of Oracle storage concepts is beyond the scope of this article. In short, though, a database is composed of Tablespaces which can contain multiple DataFiles. A Tablespace is composed of 1 to *n* DataFiles. Each of these DataFiles contains the actual database information for tables, views, stored procedures, etc. Typically, the data is segregated in such a way that system-related information is stored in a different Tablespace/DataFile than application-related data. Since DataFiles are fixed in size at database creation time, DBAs must monitor the available space and add or expand the DataFiles before they run out of room. Newer versions of Oracle, by the way, have more sophisticated space management techniques which alleviate some of these problems.

Figure 3 shows a list of Tablespaces in the database and how much free space remains. Orac has summed the total space for each Tablespace. In other words, if a Tablespace is composed of three DataFiles, then the total space available in the three files is displayed. This brings up another great feature of Orac. Each report includes a button called "See SQL" that displays the exact query run to generate the report. If there is ever any question about how a report was generated, you can get to the actual source quickly and make the needed improvements or corrections.

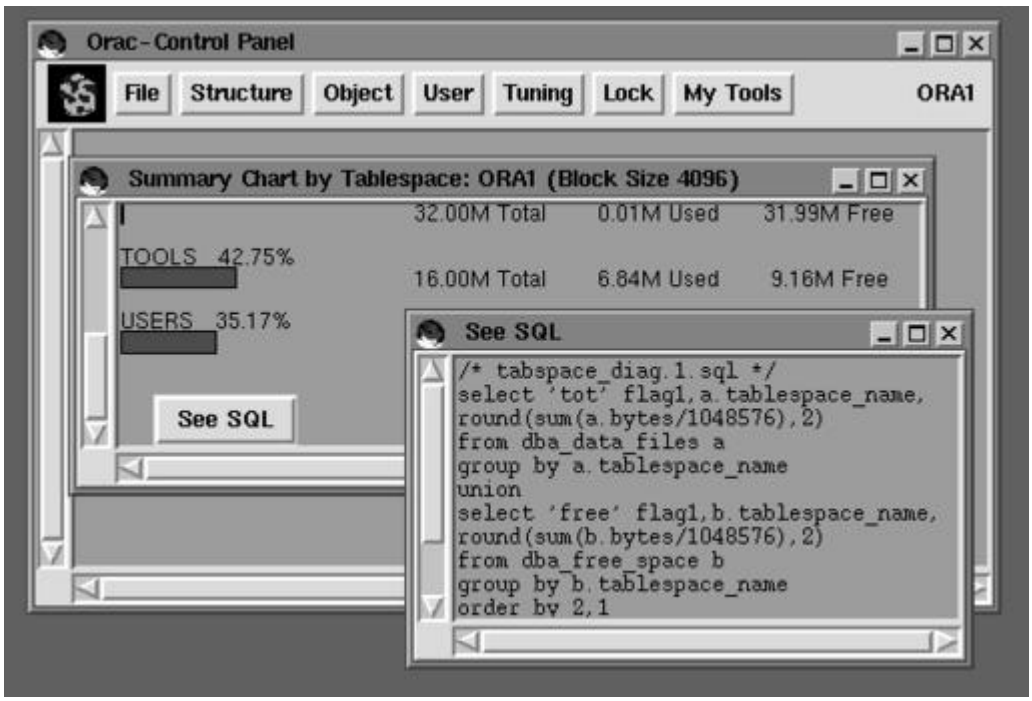



Figure 3. List of Tablespaces

## Future Directions

As mentioned earlier, Orac loads both the SQL and its user interface from a text file at startup. Orac is perfectly capable of loading a user interface and the related SQL for databases other than Oracle. In fact, developers are hard at work on Informix, and some work has also been done for Sybase. The Orac

team would very much like to see additional databases such as MySQL, mSQL and PostgreSQL supported in the future, and we're actively looking for volunteers to help out.

Another area developers are hard at work on is the **dbish** (database interface shell). This module provides the user with a way to enter ad hoc SQL into the database. The initial module has already been coded and is being tested now. By the time you read this, most of the bugs will likely have been worked out.

While parts of Orac make use of Tk to draw some primitive graphs, there is certainly room for improvement. In the near future, Orac will make use of the functionality in the **GD** and **GIFgraph** Perl modules to provide better charting and graphing capabilities.

These are only a few of the areas where work is in progress. The Orac team is actively soliciting feedback from anyone and everyone who would like to make Orac a better program.

Resources

**Kevin Kitts** is the Senior Oracle DBA at the Howard Hughes Medical Institute in Chevy Chase, MD. In his spare time, he enjoys working with Linux software including Perl, DBI/DBD and Tk and converting MS Access databases to Oracle web applications on Linux.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL



# Performance Comparison

**Tim Newman**

**Timothy S. Newman**

Issue #67, November 1999

A look at computational performance for an application running on an x86 CPU with Linux and Windows 95/98/NT, and how they compare.

Many individuals in the Linux community periodically express the belief that Linux “seems to be a faster environment” than Windows 95 and 98. However, it's difficult to find hard confirming evidence about Linux's speed, and little comparative performance data is readily available. As a result, many people considering adoption of Linux are probably unable to fully consider its performance in their decision. Of course, performance is only one of many factors that should be considered in evaluating an environment's suitability. For example, cost, reliability, usability and scalability are also important considerations.

In this article, we'll compare the computational performance for an application running under Linux and under Windows 95 and 98. We also examine if the application's performance under Linux can match performance under Windows NT. Last, we consider if the performance of Linux on the leading x86 CPU makes that environment a viable alternative to Linux on Intel.

We hope these test results provide some much-needed evidence of the potential for excellent computational performance in the Linux environment.

## Measuring Performance

A number of measures have been used to compare CPU and system performance. Two well-known benchmarks for measuing raw CPU computational capability are SPECint and SPECfp. Overall system performance has been measured using a variety of techniques, such as POVray rendering (frame) rates, Quake frame rates, Business Winstone ratings, etc. The system

performance measures are especially useful to determine how fast a particular system will be for an end user of canned applications. However, the system measures are not as useful for predicting performance of applications developed by the user. Most of the popular benchmarks tend to gloss over the joint impact of the operating system and compilation tools on execution time. Additional factors, such as the particular mix of computations, system hardware and compilation settings also impact end performance and are not isolated by most of the popular measures.

Therefore, our goal is to look at the impact of the operating system (and compilation tools) on computational performance. By isolating these factors, we can consider if Linux “measures up” to the performance of commercial Windows environments.

### The Test Environment

We tested the performance of a reasonably intensive C program which we developed and use in-house. (This program can be downloaded from ftp.linuxjournal.com/pub/lj/listings/issue67/3425.tgz.) The program implements a volume visualization technique on a medical data set. This application has moderate computational and memory requirements—the data set used in our tests is approximately 4.5MB in size and requires in excess of 300,000,000 arithmetic operations in C (most of which are floating-point calculations) to compute the visualization. (See Figure 1.) An application of this type is a reasonable test of the computational performance enabled by the operating systems and compilation tools.

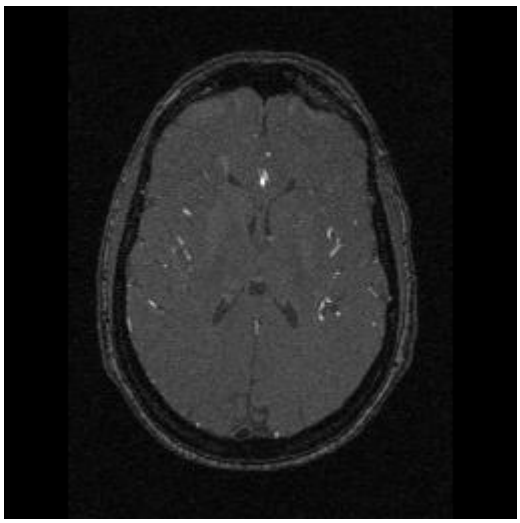


Figure 1. Sample Slice of Data

The visualization is static (no animation is involved), so we'll consider only the time to compute the visualization. The time to display the final image (see Figure 2) doesn't vary much between machines and invariably depends somewhat on the difference in graphics hardware, which is not one of the

factors we wanted to consider. We also do not consider data input and output times; our goal is to see how efficient the operating systems and compilers are for computations.

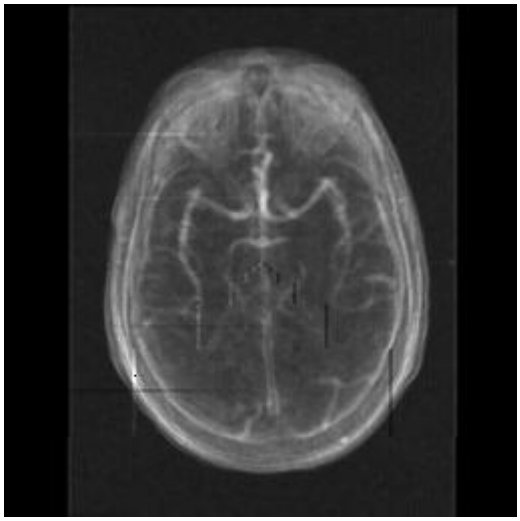


Figure 2. Rendering of Data by Program

As much as possible, we wanted to concentrate in isolation on factors that would allow us to determine which combination of operating system and compilation tools produced the best performance. Luckily, our lab has a few dual-boot machines (i.e., multiple operating systems are installed on these machines and the desired OS can be selected upon boot-up). These machines are particularly useful for performance testing, because the same hardware is used for both Windows and Linux. However, to provide a larger set of data, we also looked at performance on several single-boot PCs.

During our tests, we ensured that no other non-OS tasks were running on the computers. Although there is some evidence that UNIX in general may exhibit more graceful degradation in performance under increasing system load than Windows 95/98, it is challenging to duplicate comparable loads across different machines, so we'll concentrate mostly on unloaded performance. The tests were conducted immediately following system reboot and the average of the first three runs immediately following reboot are reported. Computation time was determined using the standard C **clock** function which returns process CPU time (at least under Linux—later, we'll discuss a bug in the clock of many Windows environments). To ensure the most optimistic measure of time, we've launched the application using several mechanisms and reported only the best time. Sometimes an application under Windows runs fastest within a compilation development environment, but in other cases, an application will run fastest directly from a command-line prompt. We report whichever produced the fastest execution time.

The following computers were utilized for the tests:

- PC 1: Pentium II/233MHz with 96 MB 66MHz SDRAM, 4.3GB Ultra DMA hard disk and dual-bootable to Windows NT or Red Hat Linux 5.2.
- PC 2: Pentium II/400MHz with 128MB 100MHz SDRAM, 9GB Ultra DMA hard disk and dual-bootable to Windows NT or Red Hat Linux 5.2.
- PC 3: AMD K6-2/300MHz with 64MB 100MHz SDRAM, 4.3GB Ultra DMA hard disk and dual-bootable to Windows 95 OSR 2 (with Ultra DMA disk drivers) or Red Hat Linux 5.2.
- PC 4: Pentium II/350MHz with 128MB 100MHz SDRAM, 6.4GB Ultra DMA hard disk and Windows NT.
- PC 5: Pentium II/350MHz with 128MB 100MHz SDRAM, 6.4GB Ultra DMA hard disk and Windows 98.
- PC 6: Pentium II/450MHz with 128MB 100MHz SDRAM, 6.4GB Ultra DMA hard disk and Windows 98.
- PC 7: Pentium II/400MHz with 256MB 100MHz SDRAM, 2 x 8GB Ultra DMA hard disks and Red Hat Linux 5.1.
- PC 8: Pentium II/350MHz with 64MB 100MHz SDRAM, 4.3GB Ultra DMA hard disk and Windows NT.
- PC 9: Pentium II/400MHz, identical hardware to PC 2 but bootable only to Windows NT.

### Compilation Tools

Several compilers were used to build executables for testing. In the Windows environment, we used Microsoft Visual C/C++ Professional versions 4.0, 5.0 and 6.0. In addition, Metrowerks CodeWarrior Professional versions 3.0 and 4.0 were used. In the Linux environment, the GNU C compiler (gcc version 2.7.2.3) and the Pentium GNU C compiler (pgcc version 2.91.57, based on the experimental GNU compilation system egcs version 1.1) were used.

To ensure the most optimal performance for the application, we exhaustively tested execution time using a large number of combinations of compiler optimization options. In this article, the reported execution times reflect the most optimal combination of optimization options for each compiler. In addition, each test was run on multiple occasions to reduce the risk of background network or computer activity producing suboptimal results.

For the Windows system, we typically experienced the best performance immediately after reboot; thus, we conducted the tests upon reboot. The Windows 95 and Windows 98 machines typically executed 2 to 5% slower before reboot. On NT machines, the time differential was less; performance was between 0 and 2% slower before reboot. On the Linux systems, we did not

experience any noticeable difference in execution times between freshly booted systems and systems that had not been rebooted recently.

### Performance Tests—Linux vs. NT

The first tests compare performance of an application built and run in the Linux environment with that of one built in the NT environment.

Table 1.

Table 1 compares the application's execution times on dual-boot machines from **gcc** and **pgcc** compilation under Linux with the performance using Microsoft Visual C and Metrowerks CodeWarrior compilation under NT. The application executed with efficiency when compiled with Code Warrior 4.0 or pgcc. For the Pentium II/400 system, the application ran fastest when compiled using pgcc under Linux. Although the program ran next fastest when compiled with Code Warrior 4.0, the GNU C compiler produced a faster executable than Code Warrior 3.0. Code produced by Visual C executes considerably more slowly than code produced with the free Linux GNU tools or the commercial Metrowerks product. Based on the PII/400, Linux with pgcc allows approximately 2% faster execution than NT with Code Warrior 4.0. However, on the PII/233, NT with Code Warrior 4.0 was barely faster (0.5%) on a freshly booted system. It is interesting that the application ran 27 to 35% faster after being compiled under Linux with gcc than when it ran under NT with Visual C. The pgcc-compiled application ran 30 to 50% faster under Linux than the same application compiled under Visual C on NT. We also found it interesting that for the PII/233, better performance was achieved using Visual C version 4.0 than when Visual C version 6.0 was used.

On the PII/233, the Visual C-compiled binaries ran about 1% slower before reboot than after. For example, the Visual C 6.0 binary executed in 36.03 seconds on a system that had been up for several hours and in 35.62 seconds on a system that had been freshly booted.

Therefore, it appears that freeware compilation tools on Linux can produce code that runs, under general conditions, at least as fast and possibly up to 4% faster than code produced using a very efficient commercial compiler for NT. Clearly, Linux with gcc and pgcc can produce code that runs remarkably faster than code compiled using the Visual C tools under NT or 95.

### Compilation Settings

We will report the most optimal execution times we could achieve. For the Visual C compiler, the fastest execution times were achieved using the maximize speed (/O2), multithread and Pentium code options in Release mode.

The maximize speed option includes inline expansion, global optimization, intrinsic function generation, frame-pointer omission and several other optimizations. There was little overall difference in execution times for the three versions of Visual C, although Visual C 4.0 code tended to execute slightly faster than Visual C 5.0 or 6.0 on the majority of the machines in our test.

For the Metrowerks Code Warrior C compiler, the fastest executables were achieved using the “Maximize Speed” optimization for a Pentium II target. CodeWarrior allows the developer to select if MMX instructions should be generated when possible, but MMX instruction generation produced well under a 1% difference in our application's execution time. We've reported the fastest time we could achieve for a given computer; if the fastest time was with MMX instruction generation, that is what is reported.

Both Code Warrior and Visual C's clock functions do not conform to the C standard. Instead of returning the process' CPU time, these compilers' clock functions return the total CPU time for all active processes. Visual C provides a **GetProcessTimes** function in NT; however, that can be used to find the actual process CPU times. Our reported times from Visual C on NT are based on the output from the GetProcessTimes function. However, since we made sure no non-operating system processes were competing for the CPU during our tests, it was typical for the CPU times from GetProcessTimes to be only 0.1 seconds less than the time reported by clock. The clock function in gcc and pgcc conforms to the C standard. We also examined wall-clock time using C's **time** function and noticed that the times reported by clock were all within a second of time's output. **time**'s resolution is in seconds, so this is the best result possible. Adding to our confidence that the application was consuming 99% or more of CPU resources while executing was a separate set of tests using WinTop, top and the NT task-monitor tools, which all consistently reported that our application was consuming 99% of the CPU resources.

For the GNU C (gcc) compiler, optimization level 2 (**-O2**) produced the fastest executables.

For the Pentium gcc (pgcc) compiler, optimization level 3 (**-O3**) with fast-math and function inlining (**-ffast-math -finline-functions**) gave the fastest execution times.

For comparison, we also ran the application on a single-boot Pentium II/400 (PC 7) running Red Hat Linux 5.1. The execution time using gcc was slightly better (14.67 seconds) on this machine than it was on PC 2. The application's execution time when compiled with pgcc was similar (13.79 seconds) to the time for PC 2.

## Comparison to Windows 98

We also tested our application's performance when run under the Windows 98 operating system. PC 4 and PC 5 have identical hardware components (e.g., the motherboards, memory, video cards, etc. are identical) except that PC 4 runs Windows NT and PC 5 runs Windows 98. Our application tends to run about 15% slower under Windows 98 than under NT. These results are summarized in Table 2. Therefore, although the application can run approximately as fast on an NT machine as under Linux (provided the program was compiled using CodeWarrior 4.0 for NT and the NT machine had been rebooted fairly recently), Linux appears to be decidedly more efficient than Windows 98.

Table 2.

We also ran the application on PC 6, a Pentium II/450 with Windows 98. Then, we applied linear regression to all of the execution times for the various compilers under all of the operating systems to extrapolate and interpolate prediction of the likely performance on newly booted systems across the Pentium line. Table 3 summarizes these results.

Table 3.

## Linux on Clone CPUs

One question that has interested us for a long time is whether Linux on “clone” CPUs is a viable alternative. We have a dual-boot machine that has an AMD K6-2 CPU and decided to test its performance for applications run under Linux and Windows 95. The K6-2 microprocessor supports 3DNow! instructions that can enable faster performance for certain floating-point computations. However, most compilers cannot generate the 3DNow! instructions, so the 3DNow! capability is usually not fully exercised. In fact, to our knowledge, the Code Warrior C compiler is the only compiler that will attempt to generate 3DNow! instructions. (gcc and pgcc do not exercise the 3DNow! capabilities of the K6-2.)

Our machine (PC 3) has a 300MHz K6-2 CPU. The machine runs both Red Hat 5.2 Linux and Windows 95. Table 4 summarizes execution times on this machine when our application was compiled and run in the different environments. We tested execution times for Code Warrior binaries produced with and without 3DNow!-optimization, and found the 3DNow!-optimized program ran about 7% faster than the non-3DNow! optimized program. Otherwise, we found that using nearly the same compiler optimization settings we had used for the Intel machines produced the best K6-2 performance. The fastest execution time was achieved using pgcc with level 4 optimization. Since pgcc does not generate 3DNow! instructions, the x86 instructions it generates

can be run faster under Linux than the best-optimized version can be executed under Windows 95.

Table 4.

Another way to look at our results is to compare the K6-2's performance under Windows and Linux with the performance of Pentium II machines. By applying the same linear regression approach we used above, we find that pgcc under Linux is the most efficient at exploiting the K6-2. Using Linux, our 300MHz K6-2 performs similarly to the predicted performance of a pgcc binary on a Pentium II clocked at 270MHz. That is, our K6-2/300 ran the application faster than a Pentium II/266 would be expected to run it under Linux. Table 5 shows the predicted clock speed of the Pentium II that would execute our application in comparable time using the various compilers. The Windows-based environments could not unlock the potential of the K6-2 as well as pgcc and Linux.

Table 5.

So, is the K6-2 a viable alternative to Pentium II, especially for Linux users? The answer would appear to be yes. Using pgcc in Linux seems to allow the K6-2 to perform like a Pentium II running at a clock speed that's about 10% slower than that of the K6-2. Considering the price differential between a K6-2 CPU and motherboard combination, Linux on the K6-2 can be attractive in terms of both price and performance.

### Linux, NT and Loading

The final experiment we conducted was to look at the total wall-clock (elapsed) time to execute multiple copies of our application simultaneously. This test is a measure of the scaling ability of Linux and NT and implicitly tests basic operating system efficiency at operations such as context switching and memory management, including cache management. We found that both Linux and NT scaled well over the tested range—up to 12 identical copies of the computations running at once. Under both Linux and NT, all the tasks began and ended computation in essentially the same pattern, and average computation time remained fairly stable.

On the NT side, we conducted one set of trials using applications compiled using Visual C 5 and two other sets of trials using applications compiled under Code Warrior 4. Two sets of trials using pgcc were also conducted. Table 6 summarizes the most strenuous tests, 12 jobs running at once. The results in the table for pgcc and Code Warrior reflect the mean results, although there was essentially no difference in measured times for the trials. It is interesting to note that under Linux, the average time per run for the 12 simultaneous

executions was only 1.2% higher than the CPU time to run one job. Under NT, the average time per job increased 2.9% for the Visual C binaries and 7.5% for the Code Warrior 4.0 binaries. It is particularly interesting that on the tested machine, a Pentium II/233—which, according to our linear regression model, is one of the only machines where NT was predicted to execute the application faster than Linux would—the total time to execute 12 jobs at once under Linux is 5.5% less than the total time to execute the jobs under NT.

Does Linux scale well? These results suggest that Linux scales very well indeed and enjoys a performance advantage over NT.

Table 6.

## Conclusions

Is Linux a viable alternative to Windows in terms of computational performance? Our results indicate that a substantial application compiled with freeware compilers under Linux can probably be expected to execute no more than 1% slower than when compiled and executed under one of the most efficient Windows NT commercial compilation systems. Under general usage (i.e., on a machine that hasn't been rebooted recently), it may be possible for an application to run up to 4% faster under Linux than under NT. Linux seems to scale very well, and under heavy loads, it appears to invariably outperform NT.

Our tests also show that Linux appears to allow approximately 15% better performance than Windows 98. The benefit of Linux is even more clear-cut when performance of applications compiled with pgcc and gcc is compared to performance when compiled with Visual C under Windows 95/98/NT.

Linux on non-Intel CPUs, such as the AMD K6-2, also appears to be a very viable alternative to Linux on Intel's Pentium II. Although reasonably sized applications can apparently execute faster on a Pentium II than on a comparably clocked K6-2 system, such applications will still run faster on the K6-2 with Linux than on the next lower clock grade Pentium II with Linux, i.e., Linux on a K6-2/400 would probably deliver marginally superior performance to Linux on a PII/350. Another way to look at the K6-2 performance is that using pgcc and Linux on a K6-2/300 would produce a program that ran only 2% slower than if it were built using Visual C and NT on a PII/400. Linux appears to be a very viable option on AMD.

For users who develop their own code, especially when that code involves a substantial amount of computation, Linux delivers outstanding performance.

**Jonathan Bush** (jbush@cs.uah.edu) is an undergraduate student at the university and a National Science Foundation Undergraduate Research Experience Scholar.

**Timothy S. Newman** (tnewman@mailhost.cs.uah.edu) is an Assistant Professor of Computer Science at the University of Alabama in Huntsville. When he's not teaching, he can usually be found working on visualization and imaging research.

Archive Index Issue Table of Contents

Advanced search

# PHP Version 4

**Craig Knudsen**

Issue #67, November 1999

From personal tool to a popular Apache module, it is now in version 4 beta.

If you've never heard of PHP, you're certainly not alone. However, you've probably visited one of the many sites driven by PHP, such as the widely popular Linux application portal Freshmeat.net.

PHP is a server-side HTML-embedded scripting language that allows developers to build web applications. PHP's features are comparable to Microsoft's ASP (Active Server Pages) and VBScript. Both PHP and ASP are interpreted scripting tools that allow HTML and code to be mixed in the same file. Both have effective APIs for building database-driven applications. However, unlike ASP, PHP is open source and cross-platform. PHP can run on Windows NT under Microsoft's IIS web server or on any UNIX variant as an Apache module or CGI.

PHP, which stands for “Personal Home Page”, was originally developed by Rasmus Lerdorf as a macro tool for tracking visitors to his home page. It grew from its simple roots into a complete tool for building web applications by version 3. Released in June 1998, PHP version 3 (often referred to as “PHP3”), was primarily developed by Andi Gutmans and Zeev Suraski.

Since then, PHP has continued to grow in popularity. It's not possible to determine exactly how many PHP-powered sites there are, but NetCraft's survey data suggest that over 500,000 web sites are running PHP. Additionally, E-Soft's Apache module survey shows over 100,000 PHP-enabled Apache servers. E-Soft anticipates PHP will become the leading add-on module for Apache, ahead of extensions for FrontPage and **mod_perl**.

## PHP 4.0 and Zend

Andi Gutmans and Zeev Suraski began working on version 4 in November 1998. Version 4 is a complete rewrite of PHP's scripting engine. Unlike PHP 3.0, the

new “Zend” scripting engine bundled with PHP 4.0 is a separate product. Eventually, Zend will be used for other scripting applications. (Incidentally, the name “Zend” is derived from the the developers' names: Zeev and Andi.)

The most significant change in version 4 is a dramatic increase in speed. The Zend team posted preliminary benchmarks of PHP/Zend and ASP/VBScript. Due to the similarities in concept, ASP is considered PHP's greatest source of competition. The various test cases show PHP/Zend running significantly faster than ASP on identical hardware under Windows NT/SP4 and IIS.

Performance was not the only design issue with Zend. Asked what other design goals he had for PHP/Zend, developer Zeev gave the following list:

- Complete platform independence—in both operating system and web server.
- Modular design—PHP 4.0 is well split into components that communicate with each other through standard interfaces (compiler, executor, web server abstraction layer, function modules, etc.)
- Stability and scalability (an almost direct result of the modular design).
- New extendibility through object-oriented (OO) overloading support. For example, PHP 4.0 natively supports OO syntax for accessing Windows COM objects and is likely to support CORBA components in the future.

### New and Improved

PHP 4.0 includes a variety of new features. A command-line interactive debugger will be included that supports break points and stepping through scripts. The debugger has not yet been released as of PHP 4.0 Beta 2.

Support for Windows COM objects has also been added, giving developers access to the wide variety of available Windows server components.

Because of the new automatic resource deallocation, resources don't need to be freed within the script, since they are freed as soon as they are deallocated. This is similar to Java's garbage collection. Note that PHP 3.0 will also free resources automatically, but not until after the script is finished processing.

New syntax changes include support for a **foreach** loop. Additionally, the terms “true” and “false” are now predefined. Pointers can now be implemented allowing two variables to represent the same variable.

New output buffering allows developers to abort a page after processing has begun. This simplifies proper error handling in scripts.

## The Roadmap

In March of this year, a closed beta was released to PHP developers and a handful of other sponsors. The first public beta of PHP 4.0 was released in July, followed by Beta 2 in August.

Feedback from the first beta release was very positive. The official word from Zeev Suraski on the release date for 4.0: “We'll release PHP 4.0 once it is stable enough for users.” The remaining work is checking all the PHP modules to make sure they're PHP 4.0-compatible. After this is complete, PHP 4.0 should be ready for release. Most likely, this will be before the end of the year.

Resources

**Craig Knudsen** (cknudsen@radix.net) lives in Fairfax, VA and telecommutes full-time as a web engineer for ePresence, Inc. of Red Bank, NJ. When he's not working, he and his wife Kim relax with their two Yorkies, Buster and Baloo.

Archive Index Issue Table of Contents

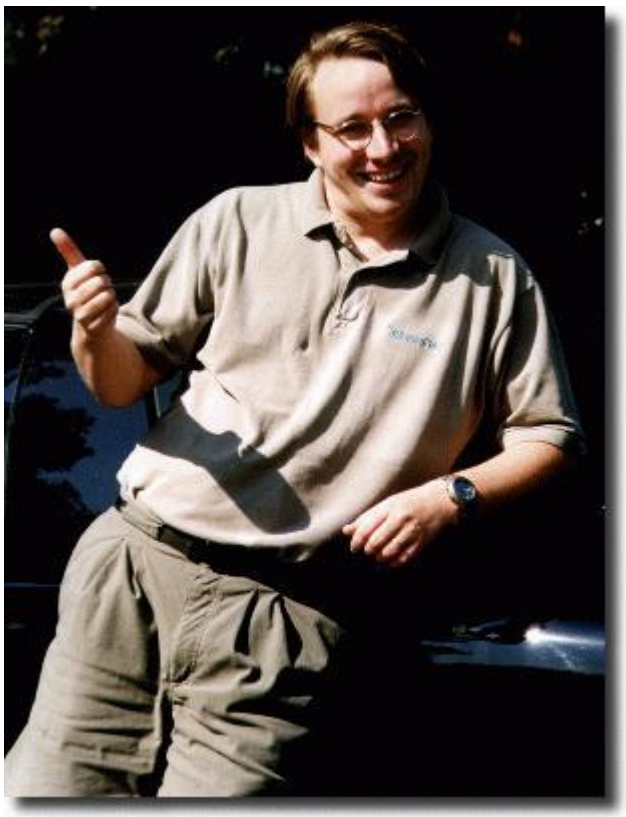
Advanced search

# Interview: Linus Torvalds

**Marjorie Richardson**

Issue #67, November 1999

A conversation with the man himself.



On August 11, while attending LinuxWorld Expo in San Jose, California, I had the privilege of having lunch with Linus Torvalds. While we had met briefly before, this was the first time I had talked to him face-to-face rather than by e-mail. He turned out to be a very personable young man—very self-assured and articulate. At his keynote the night before, he had exuded this same calm presence, answering questions with aplomb and the confidence that comes from knowing his subject well. He surprised me at lunch by admitting to being nervous about public speaking—it certainly does not show—so I guess he is human after all! I steered clear of the normal topics (Linux and Transmeta), hoping to give you a picture of who exactly Linus is. Here's our conversation.

**Margie**: Let's start at the beginning. Where were you born? When?

**Linus**: Helsinki, Finland, December 28, 1969.

**Margie**: Practically a Christmas birthday!

**Linus**: Yeah, it isn't a good time to be born if you like getting presents. You always get combination presents, but it turned out okay for me. It's a good day. In Finland, it is known as the day of “the children without fault”—not exactly “without fault”, more “almost perfect”.

**Margie**: Looks like it's true. Did you grow up in Helsinki, or did you move around?

**Linus**: I lived in Helsinki all my life, until two and a half years ago. I was as much of a city kid as you can be in Helsinki. It is not a big city—it has half a million people. I basically lived in the central parts of Helsinki all my life. I wouldn't know nature if it jumped out and bit me.

**Margie**: Oh, you never did the outdoors thing—go to the mountains, or...?

**Linus**: No, I never did the outdoors thing. Moving out here to Silicon Valley was like moving out into the suburbs—very different. I am used to living in a five-story building. Here, having a small house with a back yard feels definitely rural.

**Margie**: Well, people here don't consider living in a house rural.

**Linus**: No, I mean it's very different the way people think about living. Cities in the U.S. are much more sprawling because people want to live in their own houses. European countries are older; most people live in—not skyscrapers—but certainly buildings with five, six or seven stories.

**Margie**: How about brothers and sisters?

**Linus**: I have one sister who is sixteen months younger than I am, and I have one half brother who is nine years younger—he is twenty now. I actually have two half brothers in addition to that who are younger than my older daughter. My dad is very virile—more virile than is good for him! What can I say?

**Margie**: Well, it happens. Did you have a fairly happy childhood?

**Linus**: I'd say I had a happy, fairly normal childhood. My parents got divorced when I was small. I lived with my mother and also my grandparents. My mom was a working mother, so I ended up being at my grandparents' house for a

large portion of the time—fairly normal years. Probably what happens is you grow up to be fairly independent when you come from a family like that.

**Margie**: That could be. Your parents are both journalists, right?

**Linus**: My whole family is journalists. Everybody—my mom, my dad, my sister is getting into the journalistic thing, my uncle, my grandfather on my father's side—everybody.

**Margie**: Wow—what kind of influence did that have on you?

**Linus**: It doesn't make much of a difference because they are all different kinds of journalists. My mom was a translator; she wasn't the kind of journalist who goes out and searches for stories. She worked for the Finnish newspaper doing translations and news graphics. My dad was more of an "out there" radio journalist, so he was really excited when the Chechnyan war was going on. So he went there and had to wear a bullet-proof vest—he is that kind of person—remember, virile. My uncle works for Finnish TV.. My grandfather used to be the Editor in Chief of a newspaper. Mainly, the whole family had a kind of book-learning experience. It was considered good to know English and important to read well. So it was very natural to go to university and stuff like that.

**Margie**: Did it make you more political? Are you interested in politics?

**Linus**: I'm absolutely uninterested in politics. Probably because—I don't know—it was a fairly political family, so I may have reacted against that by being non-political. I'm not very interested. I'm much more to the left than the right in the U.S. kind of political sense. I'm fairly liberal, but at the same time, I really don't want to go into politics. My parents in the sixties were kind of radical people—they have calmed down a lot! They are not political anymore, but I grew up in a fairly political environment.

**Margie**: The sixties were pretty much like that.

**Linus**: I still meet people who say, “Oh, I knew your Dad! And I remember when you were crawling around under the kitchen table at your home when they had student meetings!” or whatever.

**Margie**: How about religion?

**Linus**: Hmmm, completely a-religious—atheist. I find that people seem to think religion brings morals and appreciation of nature. I actually think it detracts from both. It gives people the excuse to say, “Oh, nature was just created”, and so the act of creation is seen to be something miraculous. I appreciate the fact that, “Wow, it's incredible that something like this could have happened in the first place.” I think we can have morals without getting religion into it, and a lot of bad things have come from organized religion in particular. I actually fear organized religion because it usually leads to misuses of power.

**Margie**: As in holy wars?

**Linus**: Yeah, and I find it kind of distasteful having religions that tell you what you can do and what you can't do. Catholicism is an example of that kind of non-permissiveness, and I think that is very easy to get into if you are an organized religion. Religion is a very strange area. In Finland, nobody cares. Many people are religious in Finland, but it's not a political issue. Over here, religion has become politicized, so you have the fringe people in the news. And then people are afraid to talk about it because it has political implications, and that's usually not true in most of Europe. Religion is a personal matter, but does not matter for anything else. That's how I think it should be done.

**Margie**: Yes, we were founded to keep the two separated. Then the Moral Majority found out what a large constituency they had, and...

**Linus**: Yeah, it's kind of ironic that in many European countries, there is actually a kind of legal binding between the state and the state religion. At the same time, in practice, religion has absolutely nothing to do with everyday life. Maybe the taxes to the church, but that's it. They don't have any political power.

**Margie**: Here it's called tithing, not taxes.

**Linus**: Actually, in Finland they call it taxes—you pay taxes to the church. If you are a member of the church, you pay 2% tax to the church. And that's the amount of legal binding between the church and the state. Apart from that, they are completely separate. In the U.S., church and state claim to be very separate, but you still see the church has a lot of power in politics.

**Margie**: So, were you shy as a child? Or an extrovert?

**Linus**: I was an absolute geek—no social graces whatsoever.

**Margie**: None, eh?

**Linus**: None!

**Margie**: Not popular with the girls?

**Linus**: No, I can't claim I was. I was a complete nerd. I was not unpopular; I wasn't alone or unfriendly or anything like that. But I was certainly not one of the people known by everyone.

**Margie**: So you liked going to school?

**Linus**: Yeah, I liked it. I also liked the social parts. Especially at university, my studies were probably less important than being part of the student organization and going to parties and stuff like that. But especially before university, I had my own thing, which was computers and that is what I did.

**Margie**: When did you start using computers?

**Linus**: I started when I was like ten or eleven. Fairly early on, and it was early enough that you didn't have computer classes because there weren't enough people who had computers in the early '80s. I don't know, not even here probably. If you did computers, you mostly did it on your own. Maybe I had one or two other people in a class who used computers, and I met them once a

week to compare notes and stuff like that. I was never into BBSs or anything like that or the Internet, either.

**Margie**: How did you get a computer that early, then?

**Linus**: My grandfather on my mother's side owned one. He was a professor of statistics at Helsinki University. He got a computer because, to him, it was a programmable calculator. This was a home computer and not a very powerful one, but compared to a calculator at the time, it was light-years ahead. He did his own programs and had me kind of help him. Although I wasn't much help, he liked having me around. I was typing in programs—stuff like that. He probably wanted me to learn, but I didn't realize that at the time. And I was doing some playing. He wasn't interested in the games. I was reading the manuals that came with the computer and doing simple things: typing in example programs and expanding on those, like people did. It didn't start off like, the first day I saw the computer I was instantly in love. It was more like, over the years, I just spent a lot of time on the computer until it was apparent that it was my hobby.

**Margie**: Did you have a motorcycle when you were a kid?

**Linus**: Nope, I didn't have a motorcycle. I had a bike. I don't have a motorcycle now, either. I'm afraid of the damn things.

**Margie**: Are you?

**Linus**: Well, you feel so unprotected on them. They are so hard to control. Just like you have to learn to ride a bicycle, you have to learn to use a motorbike, and I haven't.

Mopeds are really big out in the countryside. That's the way people get around. Before you get a driver's license, you can still get a moped. But I always lived in the absolute center of Helsinki, so I could walk anywhere, take public transportation and whatnot. So I never got into cars or mopeds or anything like that, and I never owned a car until I moved here. Most places in Helsinki, a car is more bother than it is worth. Unless you need it for some specific reason, like transporting a kid. If we had had children then, we probably would have bought a car.

**Margie**: How about sports? Did you play football?

**Linus**: No, I didn't. I was not really bad at sports, but it was never something I found interesting. I played basketball for a while, which was ludicrous because I was always one of the shortest guys in my class. Whatever I am—5 foot 8—I'm not going to be a basketball player! I played basketball basically because my dad thought I should play a sport.

**Margie**: Oh, the virile thing again...

**Linus**: Yeah, he was more of an action kind of person. He's very intelligent, too. In fact, he is one of the more intelligent people I know, but he has his quirks. I was never much of a sports fan.

**Margie**: How about music?

**Linus**: The same. The way I listen to music still is that I put on a radio station that is basically classic rock most of the time.

**Margie**: What do you consider classic rock?

**Linus**: '60s, Beatles, whatever. Perhaps the Beatles is a bit before what I consider classic rock, but it's kind of borderline.

**Margie**: Where did you go to university?

**Linus**: Helsinki University. There are two of them: one is the Helsinki University of Technology, which is actually not inside Helsinki but just on the border outside. Then there is the real University of Helsinki, which is the kind of old-fashioned, European-like university where you have lots of courses like philosophy and languages. They do have a technical department, too, but it's not about technology. For example, computer sciences is really computer *science*—it talks about complexity analysis and things like that. I went to the real University, so I went through all the complexity analysis, what NT completeness means and the spherical side. That was kind of fun, and people always wondered why I wasn't at Helsinki University of Technology which is much more engineering-oriented. I just liked going—I also liked math, so I fit right in at Helsinki University, and at the same time, I had Linux as a side project—the practical side.

**Margie**: When did you start working on Linux?

**Linus**: Spring of 1991—eight and a half years ago. I had taken a course in UNIX and C, the fall semester before. The first time I actually touched UNIX was fall 1990, when I had a UNIX course at Helsinki University. Actually, it was the first UNIX course they ever had at Helsinki University, because it used to be a VAX and VMS place. They had just gotten a UNIX machine for trying out that newfangled thing, and it turned out to be a huge success. Within a few years, they had switched over everything to UNIX. But that first machine was used for this small course in UNIX and C, and I immediately felt that this was what I wanted to have. It made sense. Then when I bought a PC, I wanted UNIX on it, and the rest is kind of history.

**Margie**: Was there a particular professor who influenced you?

**Linus**: There were a lot of people who kind of had a large influence, but none who were really fundamental. I mean, apart from my grandfather, I usually did my own thing. I wouldn't recommend my method of study to anybody else, as it is not very organized. It basically took me eight years to get a degree.

**Margie**: It was a master's, right?

**Linus**: It was my master's, but it should not take people eight years to get a master's! But, the way I study!

**Margie**: So you were having fun, right?

**Linus**: Hmmm, yes.

**Margie**: Well, social life is important too.

**Linus**: Yeah. In Finland, you can afford to take your time because universities are essentially free, and you get subsidies from the government to study. So you can actually live on that subsidy, even though you won't be living well. I was also teaching at the university, being a T.A. at first, and then a junior research assistant, then a junior research person—slowly moving up the ranks. By the time I was starting to get my degree, I had long since realized that I actually didn't like writing papers. I had a hard time writing my thesis, and I decided I couldn't stay at the university forever. Because if you don't like writing papers, then you really don't want to stay at a university. So, that's when I said, “Okay”, and started thinking about something else, and the result was moving to the U.S.

**Margie**: Well, having computers as an early interest gives you a wide range of choices as far as what you might do.

**Linus**: Well, even two and a half years ago when Linux was new—it wasn't known like it is now; it was known only in technical circles—I had my share of interesting job offers. Most people, who get out of university in Finland and go to work, start small—Nokia is obviously very big in Finland as an employer. I had the choice of a number of very interesting and exciting places.

**Margie**: How did you decide on Transmeta?

**Linus**: The timing was good. They flew me over and gave me a tour after I signed an NDA. After the first day, I just thought these people are crazy—they were so far out. Then I went back to the hotel and basically slept on the issue. By the time I came in for the second day of interviews, I had just decided that, "Hey, if I wanted to work for a place that did something exciting, it had better be a bit crazy!" Nobody else was doing anything as exciting. At Transmeta, I had the feeling that, "Okay, this is a company that is doing something nobody else in the world is doing." And I still think that. Even though I can't say *what* it is, Transmeta is still a good company to work for. You feel that, hey, we're doing something that if it works, it will make a large difference.

**Margie**: Where did you and Tove meet?

**Linus**: We met at university; she was already a kindergarten teacher, which is her real job. She had a degree in child care, a bachelor's. She was thinking of not necessarily being a kindergarten teacher all her life, so she was doing other stuff too. And computer science was actually just an interesting sideline, so we met up at University of Computer Science. That's how we met—not very romantic—no exotic cruising in the Mexican Gulf.

**Margie**: Ahh. When did you meet her?

**Linus**: About six years ago. It was after Linux got started, but certainly long before it made a difference to anyone else.

**Margie**: Was it love at first sight?

**Linus**: No.

**Margie**: It's usually not, right? Seems to me I heard something about her being a martial arts champion or something?

**Linus**: Yeah, for six years running she was the Finnish champion in karate, but she basically stopped doing that after a while. She was into the Kata, the forms—she was doing some fighting too—but Kata is actually doing the forms of karate, correctly, at the right speed, with precision, stuff like that. The competitions she went to were basically showing what she could do.

**Margie**: What about the kids? Tell us their names and ages.

**Linus**: Patricia is the older one. She was born in Finland just two months before we moved, so she is two years and eight months. Daniella is the younger one; she is fifteen months. She was born here.

**Margie**: They are very pretty and well-behaved!

**Linus**: Yeah! They are very well-behaved, and they are used to traveling. Partly because we moved here, we didn't have grandparents to take care of them, so wherever we go, we always take them with us. So they are used to being outside with people and being at restaurants.

**Margie**: Are you planning on having more children?

**Linus**: My wife is—she is still working on me. I'm kind of resigned to having a third one. I like kids. There already will be more time between the second and third than there was between the first two. I think we should wait a bit. Life has been quite hectic with two little kids.

**Margie**: Is Tove wanting a boy?

**Linus**: Well, she always mentions it as a boy, and I bet that if she has another girl, she will start talking about a fourth one!

**Margie**: You better plan on that boy, then!

**Linus**: I am happy with two girls! Maybe when they've grown up a bit, I'll be ready for another.

**Margie**: What about school for the kids? Are y'all going to stay here in the states for them to go to school?

**Linus**: Well, that used to be kind of a major worry between us. We've seen some strange things. Tove was off looking for preschools, because you start so early here in the U.S. I looked closer at one of the papers she brought home, and found it mentioned L. Ron Hubbard. I started asking around about the place, and it turns out they are a scientology school, and they don't mention the fact that they are associated with scientology anywhere in their literature. And that kind of makes me nervous. I don't want to put my child in a scientology school by mistake. We have Patricia enrolled in a school with a good name in the area, and we will see how that goes. She is actually starting preschool this year. Assuming it all works out, we won't have to worry about school that much, not like we used to.

The difference between the U.S. and Finland is just huge in that area. In Finland nobody worries about school, nobody even thinks about it because it's obvious that you go to public school. People who don't send their kids to public school have some particular reason, like their kid needs special attention. There are a few private schools that specialize in languages and stuff like that, but it is uncommon for kids to go to those—normal people usually don't do that. Here,

you have to really think about the issue. There are a lot of bad schools and a lot of really good schools—it's more of a decision. Just that difference made us nervous, so we will see.

**Margie**: So you are happy here in the U.S.?

**Linus**: We are happier now. My wife is very happy here too, though she was not at first. She has a much closer-knit family, so our telephone bills used to be totally horrendous. We like the area. I like it from a purely technical standpoint, as well as the weather and things like that. And my wife has just grown to like the area. So we'll see. We are definitely staying here for a few more years, unless the INS puts us out.

**Margie**: Well, I doubt that!

**Linus**: Well, I don't know.

**Margie**: As long as you are gainfully employed, I thought they left you alone.

**Linus**: They do have certain rules, like for example, the visa we have is for three years and we can renew it once. In the meantime, we are trying to get a green card, and we are supposed to be real close now, but “real close now” to the INS might mean a week or four years.

This area is the worst in the entire U.S. The INS is just completely backlogged. They've had cases where they just lost files, and because they lost them, people who have been waiting for a year or two find themselves starting over with the INS saying, “I don't actually know about you.”

**Margie**: What do y'all do for fun?

**Linus**: Usually we are so busy, that we just take the kids out to nearby parks during the week. On weekends, we sometimes go to a small petting zoo close by; we are members, so we get in free. Sometimes we go to these natural reserve areas and just walk. Most of the time, we just stay home and go shopping and stuff like that.

**Margie**: You don't even go to movies?

**Linus**: We have not been to a movie in two and a half years!

**Margie**: Is this because of the children?

**Linus**: It's because of the kids. They are just now becoming the age where we are starting to say, “Hey, one of these days we could go to a kid's movie.”

**Margie**: They have a nanny, right?

**Linus**: We have a few babysitters; we get out sometimes. But again, we don't have grandparents close by who would take them for an evening at the last minute. Instead, we have to decide a week in advance to go out next Friday and get the babysitter—basically, that never happens.

**Margie**: How about fame? You seem to handle it very well.

**Linus**: It doesn't affect me much.

**Margie**: You don't get bothered a lot?

**Linus**: No. I only notice it during the conferences. Otherwise, it happens maybe three or four times a year—other than that, not at all.

**Margie**: What about all the people like me bothering you for interviews?

**Linus**: My e-mail box is a complete disaster area! It's grown over the years from about ten messages a day to a hundred messages a day. I've just gotten used to it. It can be painful; going on vacation for a week means you know beforehand that when you come back, you are going to have to spend two days just reading e-mail.

**Margie**: Do you read it all?

**Linus**: No, but I try to glance through most of it. I handle technical mail better than non-technical mail. They are much easier! You can glance and tell it's important and put it aside or handle it on the spot, and that is a fairly straightforward decision to make. But for non-technical e-mail, you say, "Hmmm...what should I do about this?" You don't have time to decide and handle it right then, so you just put it aside and forget about it.

**Margie**: I think that's the only kind of e-mail I get—the ones that need decisions. Many people in the community think of you as a hero, a role model. Who is yours?

**Linus**: I don't have any. I never had any teen idols or anything like that. I mean, if I had to pick anybody it would have to be my granddad. And then maybe more of the scientist kind of people, like Einstein, people like him. I thought that quantum physics was just the most interesting thing when I was small. But then, that was not very personalized—there wasn't just one person.

**Margie**: What kind of influence did your granddad have on you?

**Linus**: I don't know; he got me into math and computers. He was the kind of "absent-minded professor" type, so often, even when he was there, he wasn't necessarily there mentally! But at the same time, he was probably one of the closest people to me. He died when I was 15. He was there mainly during my early school years rather than later.

**Margie**: Okay, a couple of silly questions. If you could have any car in the world, what would it be?

**Linus**: Oh, God! We are actually looking at buying a car and have picked a minivan. Yes, it's not the sexy sports car. We're getting a lot of friends and family over from Finland and whenever we go somewhere, we have to take two cars and that is just a pain. So we are getting to be a "real American family", and we will have a minivan. But that was not the answer you were looking for!

**Margie**: That's okay! Would you rather have a Porsche?

**Linus**: No, I never liked that kind of muscle car—the Corvettes, I didn't like. I find those to be uninteresting—a toy car. I like the Miata, but it's too small; however, the BMW Z3—that kind of car—that, I like. But it's just too impractical when you have two kids. We have considered maybe getting a convertible. When we moved here, we didn't think about that, but actually in California it does make sense much of the time. But we have to get a four-seater anyway, so even then it's not going to be one of the really "fun" cars; it's going to be one of the practical ones.

**Margie**: Do minivans come with sun roofs?

**Linus**: Maybe sun roofs, yes, but not convertible!

**Margie**: What's your favorite color?

**Linus**: It was blue when I was small—the classic boy color. I think it's yellow now. Yellow is a hard color, okay? You can do it too light and it just looks pee-colored. Or you can do it too dark and it becomes too orange, so you have to be careful—blue is safer.

**Margie**: What message would you like to give our readers?

**Linus**: Linux didn't start out as a message to the masses. Unlike Richard Stallman, I really don't have a message. He has one and can go on about it forever. I'm just an engineer. Let's see: Do things well! Do them with heart! What other strategies can I come up with...?

**Margie**: That will do. Thanks very much for taking the time to talk to me.

# ICP vortex GDT RAID Controllers

**Eric Green**

Issue #67, November 1999

While ICP is the largest supplier of RAID hardware in Europe, they appear to have a low profile here in the U.S. compared with Mylex, Adaptec and AMI.

- Manufacturer: ICP Vortex
- E-mail: sales@icp-vortex.com
- URL: http://www.icp-vortex.com/
- Price: $1595 US for GDT6518RD $2195 US for GDT6538RD (volume pricing available)
- Reviewer: Eric Lee Green

ICP Vortex is a German company that has been a long-time supporter of Linux. The ICP driver was written by ICP and has been included in the Linux kernel since the Linux 1.3 days, and their RAID configuration utility (**gdtmon**) runs natively under Linux. While ICP is the largest supplier of RAID hardware in Europe, they appear to have a low profile here in the U.S. compared with Mylex, Adaptec and AMI.

What ICP provides is a number of RAID cards ranging from a low-cost one-channel RAID0,1 controller to a series of high-end fibre-channel controllers. All ICP cards work with Linux and are supported by current Linux distributions “out of the box”, with the exception of Red Hat 6.0, where the ICP driver was inexplicably left off the boot disk despite being on the install menu. A fixed boot disk is available from Red Hat's FTP site or directly from ICP.

Screen Shot

First, a bit of background: RAID is a method for combining disk drives for either performance or reliability purposes. There are a number of approaches to doing RAID. Software RAID is built into the Linux kernel. “Pure” hardware RAID has the SCSI controller(s) actually built into the PCI card. There are also hybrid

approaches that have the advantage of being cheap but requiring specially equipped motherboards (usually with what is called a “RaidPort”) or double the PCI bandwidth in order to run.

The biggest advantages of “pure” hardware RAID are low CPU usage, robustness, minimal bus bandwidth usage and avoiding the underlying limit in the number of SCSI devices that can be addressed by the Linux SCSI driver architecture. The Linux SCSI subsystem can access a maximum of sixteen hard drives (labeled sda through sdp). By presenting the multiple drives in a RAID array as a single drive to Linux, hardware RAID bypasses that limit.

## Evaluation of RAID Controllers

The first problem was deciding which ICP RAID controller to test. It was easy to discard the low-end and high-end controllers. The 61*xx* series, which does only RAID0 and RAID1, is slower than the built-in Linux software RAID. The fibre-channel controllers work well under Linux using the EXT2 file system, but Linux support for file systems capable of using the fibre-channel drives as fibre-attached storage (shared between multiple machines) is in experimental stages at best.

To decide which to use, I turned to the two most common uses of RAID under Linux: web and news servers.

Web servers typically have three drives, a hot spare. With 4.5GB SCSI hard drives, this would give them 9GB of usable disk space. Note that 9GB SCSI drives are generally twice as fast as 4.5GB, so that would also be an option. A one-channel 6518RD works fine in this application.

News servers need massive amounts of disk space. A news server capable of backing up the entire USENET news hierarchy for a month will probably need at least twelve 36GB hard drives. My informal benchmarks show that an IBM 7200rpm 36GB hard drive transfers data at approximately 25MB/sec, while Ultra2 SCSI has a bandwidth limit of 80MB/sec. Thus, four drives per channel would be a good “seat of the pants” estimate here, requiring a 3-channel RAID controller. The 6538RD, one of their “mid-range” controllers, was chosen to be the second card evaluated.

## What I Got

The ICP controllers come in a fairly hefty box containing the controller, a CD with drivers and utilities, and a hefty wire-bound manual. The CD also contains a full PDF version of that manual as well as any addendums and errata created since the manual was published.

Examining the 6518RD uncovered an Intel 960 CPU, a Symbios 53c895 SCSI chip set, a Symbios 53c860 chip and various small support chips. There is also a socket for cache SIMM. This also serves as the Intel 960's scratch memory, so the controller will not operate without it.

The 6518RD and 6538RD both contained the 53c860 Ultra-SCSI chip to handle CD-ROM and tape drives on a separate narrow-SCSI bus. Thus, the 6518RD can actually be considered a two-channel device, though it has only a single Ultra2 RAID channel. Despite having two channels, the 6518RD has a single external SCSI connector on the back. Annoyingly, this connector goes to the Ultra2 controller. Note that attaching any non-LVD devices to an Ultra2 bus will slow it down to non-LVD speeds, i.e., 40MB/sec, so hooking that tape drive to this connector would be a serious mistake. An external tape drive will require a special cable bringing out the internal narrow SCSI to a socket on the back of the computer.

The 6538RD looked similar to the 6518RD, but was longer in order to hold the two additional 53c895 chips. It has three of the high-density SCSI connectors on the back. Similar to the 6518RD, only the Ultra2 channels are brought out to the external connectors. The workaround if you have an external tape device is the same: bring the internal narrow-SCSI connector out to an external connector using a special cable.

## Making Them Work

The first thing I had to do was add the cache SIMM to the controller card. ICP recommends you use 60ns FPM (Fast Page Mode) RAM or 50ns EDO RAM. I had only 60ns EDO RAM, so at first I tried setting the jumper to say I had 60ns FPM RAM. After talking with technicians, I decided to move the jumper to EDO mode. I discovered a significant performance difference between FPM and EDO mode settings. The jumper to EDO mode increased performance by approximately 20%. Using a cheap 60ns EDO SIMM definitely made the controller unreliable; in fact, it corrupted the hard drive when I tried copying multi-megabyte files to test disk write throughput. Switching to a good-quality OEM 60ns EDO SIMM solved that problem.

I tried benchmarking the GDT with both 64MB and 128MB of cache. I found no significant difference in performance, and thus recommend 64MB for cache. Given the low price of RAM today, it does not make sense to use less than 64MB for cache.

Both cards work identically when you put them into your computer. They use the same driver, the same BIOS and the same utilities. The only difference is that the BIOS utility you use to set up your RAID volumes shows more channels for the 6538RD.

The BIOS setup utility allows you to select drives and then combine them into a single RAID volume. It does not allow dividing a drive between multiple RAID volumes as is possible with the software RAID driver. The setup utility writes the resulting data into a special boot sector at the start and end of the drives. Thus, you can remove the controller, put in a different (replacement) controller, and your RAID setup remains the same.

The GDT6538RD had no trouble combining drives from multiple channels and presenting them to Linux as a single SCSI hard drive. Curious, I tried putting multiple GDT controllers into a machine to see if I could combine drives which were on entirely different controllers. This did not work, though otherwise the Linux GDT driver had no trouble with handling multiple GDT cards in the same computer.

Once the array was configured, the GDT controller started building the array, i.e., building the checksum blocks. I interrupted this process to reboot into the Red Hat 5.2 installation routine. I discovered the ICP does not present a SCSI CD-ROM hooked to its Narrow SCSI port as a bootable device to the BIOS. Swapping to an IDE CD-ROM solved that problem.

The 5.2 installer detected on my system, an ICP RAID Array Controller and the RAID array as a single hard drive. I went ahead and installed Red Hat Linux. While I was doing this, the GDT controller was continuing to build the disk array, transparently, in the background.

It can take quite some time for the arrays to build and become redundant. Note that you can go about the task of installing the OS, configuring your software, etc. while the array is building in the background.

Screen Shot

## Performance

Unfortunately, I was not able to do extensive benchmarks on the system with the 3-channel controller and 36GB drives. The command **hdparm -t** reported 28MB/sec throughput on “virgin” drives (where the OS had just been rebooted and the GDT controller reset). Using **dd** to write 100,000,000 bytes from /dev/ zero to the disk array reported a write throughput of around 18MB/sec. One thing I did discover was that turning on the write caching sped up throughput considerably. Apparently this allows the controller to do write re-ordering internally and combine writes when possible. The tested 2.0.36 and 2.2.10 kernels both properly flush the cache at shutdown time, so as long as you have a UPS that is properly configured to do a clean shutdown of the system, this is fairly safe. If you don't trust the UPS software and insist on turning off the write cache, expect the write performance to be significantly impacted.

The theoretical performance of the hardware involved was somewhat higher than the numbers seen. The EXT2 file system was eliminated as a possible factor by the expedient of using dd to read and write to raw partitions. Software RAID0 was faster by about 15%, but still did not approach the theoretical performance of the hardware involved. Speculations on the cause of this slowdown would be interesting (I suspect they happen due to various factors within the Linux kernel), but are irrelevant to this article. The GDT's RAID5 performance, in any event, performed similar to the software RAID5, without the excessive CPU usage seen while running the software RAID5.

### Safety

If a drive fries, RAID1 or RAID4/5/10 keeps going. The GDT then starts beeping annoyingly. It also sends a message to both **syslog** and the console.

If a hot spare was defined, the GDT will automatically mark the bad drive as failed and switch to using the hot spare. It will transparently rebuild the array with the hot spare. No action is needed on your part, though you will eventually want to remove the bad drive, replace it with a new drive and initialize the new drive as a hot spare. Assuming you have hot swap trays, you don't need to shut down Linux to do this. The ICP gdtmon program runs natively under Linux and will handle this situation.

If you have no hot spare, the GDT will automatically mark the failed disk, but the array will no longer be redundant. Again (gdtmon to the rescue), you can use gdtmon to swap out the bad drive and swap in a replacement. No down time is necessary, since gdtmon runs natively under Linux; the new drive will be transparently rebuilt while your system continues to run.

### Price

Like Mercedes-Benz cars and most other German products, the GDT is somewhat over-engineered. This makes it very reliable and safe, but also more expensive than the competition. I tried two different resellers of ICP controllers. One tried to sell me the controller and RAM at the suggested retail price, the other quoted me a price for the 6518RD (with RAM) of approximately $50 less than the suggested retail price for the controller alone. This price with 64MB of cache was approximately $300 more than the equivalent Mylex ExtremeRaid with 32MB of cache. Prices fluctuate due to new product introductions, etc., but these relative prices will probably remain similar. Thus, for low-end RAID you may wish to look at competing devices, bearing in mind that most vendors do not have the Linux experience and range of support that ICP has.

## Conclusions

The robustness, transparency and ability to present multiple drives as a single volume to the Linux SCSI layer makes hardware RAID a must for situations requiring high availability and large amounts of disk storage.

Within the hardware RAID market, the ICP GDT line stands out as the most complete set of RAID solutions available for Linux. Other vendors have single-channel RAID controllers for Linux, but multi-channel RAID controllers are still a rare bird in the Linux market. No other vendor at the time of this writing offers fibre-channel RAID solutions for Linux.

The ICP GDT line also stands out as one of the most robust RAID solutions in the Linux market. In part, this is because ICP ported their gdtmon utility to Linux, allowing handling of hot swap and failover situations without having to reboot to DOS like you must in order to reconfigure most competing devices. Much of it is because the ICP team has engineered their product to be as safe as possible.

Unfortunately, this does come with a cost. The ICP RAID controllers are not the cheapest. For a single-channel controller, you may wish to look into competing devices from Mylex, AMI or Adaptec, bearing in mind that their support for Linux is relatively recent and incomplete. For multi-channel controllers or fibre channel RAID, there is no argument at all. ICP Vortex is *the* RAID solution for Linux in those markets.

Eric Lee Green (eric@estinc.com) is the systems and networking guru for Enhanced Software Technologies Inc., “The Bru Guys”.

Archive Index Issue Table of Contents

Advanced search

# DB2 Universal Database for Linux, Version 5.2

**John Kacur**

Issue #67, November 1999

Among the impressive list of big-name databases now available for Linux is IBM's DB2.


- Manufacturer: IBM
- E-mail: ibmdirect@vnet.ibm.com
- URL: http://www.software.ibm.com/data/db2/linux/
- Price: free download or $39 US (Personal Developer's Edition)
- Reviewer: John Kacur

Among the impressive list of big-name databases now available for Linux is IBM's DB2. You can obtain DB2 by visiting IBM's web site. You can order a 60-day trial, DB2 Universal Database evaluation CD-ROM for free. There is also a Personal Developer's Edition which is not time-limited and is also free when downloaded. The downloads are rather large, however; the latest version's core download is 59MB, with separate downloads for extras such as PostScript documentation. If you don't have a fast connection to the Internet, you would be better off ordering the evaluation CD-ROM or pay for the Personal Developer's Edition CD-ROM, which costs approximately $39 US.

## Installation

The version I tested was the IBM DB2 Universal Database for Linux, Version 5.2. (Version 6.1 is now available.) It consisted of the following products:

- DB2 Universal Database Workgroup Edition
- DB2 Client Application Enabler
- DB2 Software Developer's Kit

as well as two books in HTML format directly on the CD-ROM:

- *DB2 Quick Beginnings for Linux*
- *Installing and Configuring DB2 Clients*

A wealth of documentation can be installed on your hard drive, not just for installation and configuration, but for application development and database administration.

DB2 was developed under the 5.1 Red Hat distribution, but should work under any distribution with the following prerequisites.

- Linux Kernel 2.0.35 and above—I tested on a 2.0.36 kernel, and it is reported to work on 2.2 kernel versions as well.
- A pdksh shell is required to run the DB2 command-line processor.
- rpm-2.5.5

IBM also recommends a minimum of 64MB of RAM and 128MB of swap space, as that is the minimum configuration they used for testing. I was able to function with the recommended 64MB on my server and only 32MB on my client node.

The installation process was relatively straightforward. You need to install with superuser privileges. First, mount the CD-ROM and change to the directory where it is mounted. Then enter the command **./db2setup** to start the installer program. I recommend using the **-d** flag, which will generate a trace file called /tmp/db2setup.trc, which you can examine in case anything goes wrong. A db2setup.log file will also be generated in either case. My first attempt failed because the installation program assumed that /usr/sbin was in my superuser path. This might not be the case if you log in as a normal user and issue the **su** command. You can fix this by temporarily adding this directory to your path. Under bash, for example, you would issue the command:

```
export PATH=$PATH:/usr/sbin
```

The **db2setup** program is a menu-driven ncurses-type install. You should decided before hand which components you plan on installing, depending on whether you are installing a database server or just a client.

### Going for a Test Drive

The simplest situation is when the client and server are both installed locally. You need to log on to the system as the user you created when you made a DB2 instance during the installation. This is most likely db2inst1, if you accepted the defaults. Move to the /home/db2inst1/sqllib directory and execute the **. db2profile** script if you are using a bash or pdksh shell. If you are using csh, then execute **.db2cshrc**. This will set up some environmental variables. You can append this script to your .profile file to run automatically upon login. If you haven't created a sample database during the install process, move to the /home/db2inst1/sqllib/misc directory and execute **./db2sampl**, which will create a test database called “sample”. This can take a few minutes. Now you're ready to connect to the database and interact with it. One way to do this is to use the Command Line Processor. You can do this in Command Line Mode simply by prefacing all your commands with **db2**. An advantage to using this mode is you remain in your shell. You can also enter the Interactive Input Mode, by entering the **db2**. For example, in the Command Line Mode enter **db2 connect to sample**. If everything is configured correctly, you should see this on your console:

```
Database Connection Information
 Database server        = DB2/LINUX 5.2.0
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE
```

### Do It with a Java-Based GUI

If your client is connected to the server over TCP/IP, you can use a Java applet from your browser to access the database. To do so, you need to use the Command Line Processor to catalog the remote node for the client. After that, you can open the applet in your browser by selecting Open Page from the File Menu and choosing /home/db2inst1/java/prime/db2webcc.htm.

### Conclusions

I've tried to provide enough information to get you started. One of the best things about this database is the huge amount of information IBM provides in the on-line books as well as at their web site. There were a few features not available in the 5.2 beta I tested, and these were documented in README.linux. Support from the big-name database providers removes yet another barrier to the widespread use of Linux in corporations, and DB2 on Linux looks like a winner.

**John Kacur** (jkacur@acm.org) has a B.A. in Fine Arts. After two years studying Russian in the Ukraine and another two years teaching English in Germany, John returned to Canada to pursue a second degree in Computer Science.

Archive Index Issue Table of Contents

Advanced search

# CodeWizard for Linux

**Ben Crowder**

Issue #67, November 1999

Codewizard is a tool that searches through your code for violations of good C++ coding standards.


- Manufacturer: ParaSoft
- E-mail: info@parasoft.com
- URL: http://www.parasoft.com/
- Price: $2495 US (single license)
- Reviewer: Ben Crowder

You're working on your latest C++ project, an infinitely cool piece of software that everyone is dying to see. Before you release, however, you want to get rid of as many bugs as possible. You know you have a certain bad habit you learned in C, the one you have to go back and fix time, after time since you keep forgetting you're writing in C++ now.

Enter *CodeWizard* by ParaSoft, a tool that searches through your code for violations of good C++ coding standards. Based on programming ideas from Scott Meyers' books *Effective C++* and *More Effective C++*, CodeWizard automatically enforces thirty-five descriptive rules or “items” for effective C++ programming—standards that significantly reduce the number of errors in your code. In addition to Meyers' rules, ParaSoft has included a list of thirty-six additional items.

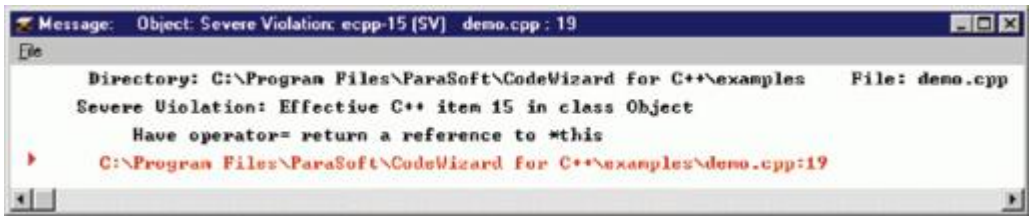



Figure 1. Rule Violation Message

What kind of rules are we talking about? First of all, the items have each been given a rating according to the severity of the rule. The five ratings, from least serious to fatal, are: Informational, Possible Violation, Violation, Possible Severe Violation and Severe Violation (see Figure 1). An example of an Informational rule is item 2 under the *Effective C++* list: “Prefer iostream.h to stdio.h.” So, for instance, if you include stdio.h in your code, one of the warnings that will come up will explain that you shouldn't use stdio.h when coding in C++ (see Figure 2). A Possible Violation is rule #17: “Check for assignment to self in operator=.” Item #22 catalogs a Violation: “Pass and return objects by reference instead of by value.” A Possible Severe Violation is something like, “Don't try to return a reference when you must return an object.” Severe violations can prove to be fatal to your program. Rule #10 is a good example: “Write delete if you write new.” Seems obvious, of course, but you may forget it and thus cause ugly memory leaks. The items are divided into different categories, such as “Shifting from C to C++”, “Memory Management” and “Inheritance and Object-Oriented Design”.

Figure 2. C++ Code that Violates a Rule

CodeWizard came with some instructions on installation and basic usage. The distribution file is a tar file—it's up to you to manually copy it over from the CD and unpack it. A graphical installation program would come in handy for less experienced users (although you probably won't be using the UNIX version of CodeWizard, if you aren't familiar with a UNIX shell and basic commands). You can configure CodeWizard as either root or a regular user—both ways worked fine for me. A license is required, which must be obtained from ParaSoft. The **pslic** program, the ParaSoft License Manager, will give you a machine ID and a network ID, which you give to ParaSoft. In return, you'll receive an expiration date and a password which will unlock CodeWizard. I'm sure there's an easier way to manage licenses, one that doesn't require contacting ParaSoft, but this method minimizes pirating (which is, of course, a concern only for commercial software vendors—the GPL is beautiful in that it completely shatters software piracy).

You use CodeWizard in place of your compiler on the command line. Instead of typing

```
g++ project.c -o project
```

you would use

```
codewizard project.c -o project
```

You can set the compiler for CodeWizard to use in a configuration file. CodeWizard then scans your code for any of the violations in its rule set. On a side note, you can suppress any of the rules (in case some of them make no sense when applied to your particular project), or you can add your own rules using RuleWizard (see Figure 3). There is a Motif GUI program, Insra (see Figure 4), that displays the error messages and allows you to open the source files, make any necessary changes, rebuild the program and retest the code for violations. If you want to keep everything strictly console-based, however, you can do that as well.

Figure 3. Rule Wizard Screen

All in all, I'd have to say that CodeWizard can be an incredible help in C++ coding. A Java version is available as well, and with the ability to add your own rules, you could foreseeably use it for almost any other language as well. Though it *is* expensive, if you're serious about your C++, this would be a wise investment.Version 3.0 is due out in September.

Figure 4. Motif Program Insra

**Ben Crowder** is a young Linux aficionado living in Utah. In addition to fiddling with the insides of computers, Ben enjoys reading, writing and music. He can be reached at mlcrowd@enol.com.

Archive Index Issue Table of Contents

Advanced search

# VA Linux Workstation VArStation XMP

**Jason Kroll**

Issue #67, November 1999

Extra, extra, read all about it.

- Manufacturer: VA Linux Systems, Inc.
- E-mail: sales@valinux.com
- URL: http://www.valinux.com/
- Price: $3,000 to $9,000
- Reviewer: Jason Kroll

The VArStation XMP (analogous to the StartX ZP) is a top-quality, high-powered workstation from VA Linux Systems. In the years since its 1993 inception, VA has earned a reputation for producing rock-solid, dependable, performance-maximizing, Linux-based workstations, servers and clusters. Once upon a time, VA also built laptops and hopefully will do so again soon. VA even has an entry-level machine, the SP, if you want to get a taste of the “good life”. The XMP lives towards the upper range of workstation machines and is indicative of a typical VA Linux system. Most VA workstations come in the same tower, which is an attractive, solid, heavy container with plenty of space and a powerful fan. Beyond that, VA allows any type of customization you want and provides an on-

line system for piecing together whichever components you like. Here's a description of the machine VA Linux Systems sent to *Linux Journal* for review.

### Hardware

The core of the system is an Intel MS440GX motherboard featuring dual Pentium III Xeon 550MHz processors with 32KB of L1 cache and 512KB of L2 cache, 512MB of RAM and an 8.4GB SCSI drive from Quantum. For comparison (to be somewhat up to date), current VA workstations come with an Ms4406x or Tyan 1832 motherboard, are available with Pentium III and Pentium III Xeon processors, and typically have 512KB of L2 cache with the option of expanding to 2MB of L2 cache. Modern machines also have larger hard drives, usually SCSI of 18.2GB or so and often 512MB or more of SDRAM. What does this jargon mean?

L1 cache is “level 1 cache”, static RAM which sits on the processor. When your processor asks for an instruction from memory, it's very likely that the next instruction it will ask for is the one right below the current instruction in memory. Likewise, when one piece of data is loaded, the next piece of data the program will want is probably sitting right below that. Instead of loading the instructions individually, the processor caches 32KB or more on the chip (usually 16KB for instructions and 16KB for data) so it can get at the instructions and data immediately, instead of loading them from RAM every single time. This dramatically improves processor speed. The larger the L1 cache, the less loading from L2 cache and the faster the operation.

L2 cache is the “level 2 cache”, static (or sometimes dynamic and static) RAM which sits between L1 and the memory at large. Normal RAM is very slow (but cheap) compared to L2 cache (which in turn is slower and cheaper than L1 cache), so the L2 cache acts as a buffer to store large chunks of memory to be grabbed by the L1 cache, except L2 grabs 512KB or more at a time (depending on the size of your L2 cache) compared to the usual 32KB grabbed by L1. L2 is often considered the bottleneck in computer performance, but VA helpfully recommends that you buy bigger cache only if you do *very* many floating-point operations and your application is noticeably limited by L2 cache dependency. L1 cache loads instructions very quickly from L2 cache, which loads instructions from RAM, so the system is a kind of filtering pyramid to make sure the most needed instructions are most readily accessible.

One might expect a multitasking OS like Linux, which frequently grabs instructions and data from all over in memory, to attenuate the benefits of cache, but the granulation of multitasking systems is large enough that the effect is minimal, especially on fast processors. (Granulation in this case is the division of tasks into pieces, since multi-processing operating systems actually

execute many lines of a task before going on to the next task in a queue, and on a 500MHz chip, many instructions and much data can be covered in a few microseconds before moving to the next task, hence the cache is definitely utilized.) Still, VA Linux recommends increasing performance by adding more memory (since we all know how well Linux takes advantage of extra memory) or using multiple machines rather than overloading on cache, as the most economical approach.

SDRAM, or synchronous dynamic RAM, is dynamic RAM which is synchronized for the system bus speed (which is different from the processor speed). That is, dynamic RAM loses its charge and constantly requires refreshing, and if the RAM refresh is synced to the speed of the processor's system bus, a noticeable performance improvement can be made. VA uses SDRAM which is the most fashionable at the moment. Although it doesn't translate into a monstrous performance increase, especially because of the very high speeds of L1 and L2 cache and the efficiency of this system, at the very least it's not a weak link. The core of hardware performance tends to be based around the CPU.

The motherboard of the VArStation contains two Intel Pentium III Xeon 550MHz chips, though one is by no means compelled to choose such a setup. At the moment, VA Linux offers a range of processors starting with Intel's Celeron 400MHz and ranging up to 600MHz Pentium IIIs and 550MHz Xeons. Workstations have one or two processors, servers are available with four, and clusters have more. The Linux kernel can take advantage of two processors and is said to do well with four, although it is rumored more might be gratuitous and a bit less efficient than distributed computing.

The difference between various chips can be confusing. Why is a Pentium III better than a Pentium II, and what do terms like MMX and Xeon mean? Chip design is actually quite complicated, especially with complex instruction set chips (CISC) like the x86, of which I'm not particularly fond. While there are issues of how a chip manages caching of instructions (and microarchitecture techniques such as multiple-branch prediction, data-flow analysis and speculative execution), much of the performance increase in the latest Pentiums and the like comes from raw speed and cache size, and differences between chip models have to do with cache and assembly-code instructions. For example, MMX means MultiMedia Extensions, which is 57 extra multimedia instructions, eight new registers, larger L1 cache and the ability to perform a single instruction on multiple data, known as SIMD. Xeon is the futuristic name given to the top of the Pentium line, designed for server applications. Xeons are made for multiprocessing and use 100MHz buses and 32KB of L1 cache via a 64-bit bus, with support for larger addresses for dealing with big machines. Xeon 36-bit addressing, for example, can access 64GB of physical memory, which may be 62GB more than Linux kernels support. At the moment, VA Linux

uses Pentium IIIs for single-processor machines and Pentium III Xeons for dual-processor workstations. Suffice it to say, III is better than II on the whole, and Xeon is better where parallel processing is concerned. The entry-level machine from VA, the StartX SP, is based on the Celeron 400MHz, which has smaller L2 cache (128KB), although it is located directly on the chip for faster results.

All in all, VA matches the right chips to the right machines so that you get maximum performance for the price. For example, a Celeron is good for a home computer, whereas a Xeon would be inefficient overkill on a single-processor desktop machine. The important thing about Linux box builders, especially VA, is that they know how to optimize systems. VA is quite knowledgeable and helpful in recommending how to get the best performance for the dollar, and there really is an advantage to getting a system that's built especially for Linux machines.

Everything else on the VA boxes is top of the line, from SCSI drives to Ethernet cards to modems. There are no weak links in VA boxes. My only complaint is that the sound cards are not spectacular (for example, I would really like digital I/O), but Linux has a way to go as far as sound support. Video, on the other hand, is top quality and utilizes the latest Matrox cards. Of course, VA Linux computers come optimally preconfigured for the monitors VA sells. The keyboard is nice and heavy, although it has three stupid Windows keys. The mouse is a Logitech (for a right-hander, I might add). You don't have to worry about getting cheap components on a VA system. I asked VA about this. “We want overall quality to be high; we will not sacrifice quality to get the world's cheapest prices,” I was told. As for support, VA has 24x7 support staff, but the manual is so thorough you probably won't need support. The manual is actually a very large, thick, technical-looking collection of hardware data for everything in the machine. In addition, VA included Que's *Using Linux* book, SuSE's Linux Office Suite 99, a rescue disk, a couple of CD-ROMs for the video card and motherboard, and a CD-ROM of the OS. The system software for VA Linux workstations is the VA Linux OS.

### Software

VA Linux OS 6.0 is actually a Red Hat-based CD-ROM which is customized for VA machines and their hardware. Although the systems come pre-installed, installation is quite easy and all one has to do is select the VA Workstation option from the package selection menu to get the VA wares. Obviously, you don't have to enter your hardware configurations, since either Red Hat is probing successfully or VA made sure the installer already knows.

VA Linux OS 6.0 differs from Red Hat in a number of ways, and in my mind, is an improvement. For one thing, VA's installation uses KDE instead of the other

window managers, and regardless of how one feels morally about the GNOME vs. KDE debate (libQt still isn't truly free), KDE is by most accounts further developed at this point and does support several languages. I especially liked VA's account of why KDE was chosen:

> The primary reasons were stability and consistency of interface. This is not a judgment against GNOME or for KDE, but merely a decision based on what is best for the novice user. It's our belief most advanced users will already have a preferred environment, which would not be GNOME or KDE in any case.

VA's variant of Red Hat looks quite nice; in fact, it's my favorite of all Red Hat variants including Red Hat itself. As far as became apparent to me, it's just a highly configured setup with a custom kernel (2.2.7-1.15smp) for the VA box. Nevertheless, the desktop looks very nice by default with a futuristic VA background and cool blue borders and buttons that look rather in tune with the whole quest to be “space age”, which comes with this millennium business.

Despite everything VA has going for it, a few small problems exist in the VA OS 6.0. Some kernel modules are compiled with an earlier kernel and require **insmod -f** to force them to load, which can lead to unresolved symbols. If you load the modules in the right sequence, you can usually get around this, although after I reinstalled the OS (the machine shipped with VA Linux OS 5.2) the sound modules remained dysfunctional without a kernel recompile. So, it seems to me that the weakest link in VA systems is audio. Nevertheless, it's not so hard to recompile a kernel for sound support, and these are exceedingly fast computers so kernel compilation goes quickly.

## Benchmarks

The *Linux Journal* benchmarks are still genuine vaporware, but as soon as they are developed, we'll print the results for this excellent VArStation. Computer systems are changing, moving from a single processor to multiple processors, supporting larger hard drives and much more RAM, and are designed for efficiency gains in areas often ignored by conventional benchmarks. In addition, greater customization of hardware can take the load off the CPU, while kernels are becoming more and more clever. As computers are whole systems rather than individual chips and pieces, the issue is not how well one of the processors performs a loop of floating-point operations, sets all the bits in memory, or scores on a bogomips test (547.23 and 545.59 for the processors on this particular VArStation), but how well the system as a whole performs. Overall system performance is dependent on many things, especially the operating system itself and how well it can manage various tasks and take advantage of hardware. Newer Linux kernels perform better.

Speaking of kernels, the most important activity of any Linux user is recompiling the kernel. Well, maybe not, when you've already got an ideal kernel as on a VA box, but it's often the biggest compilation many of us make. On a VArStation running **egcs** 2.91.66, my typically oversized kernel compiled with **make bzImage** in less than two minutes, with **make modules** requiring only three. **make modules_install** took two seconds and **bzlilo** ran in eleven. Although everyone's kernels are different, mine are usually too big, so yours would probably take even less time.

The machine is a true joy to use on a daily basis. There are no lags, no waiting, and not even Netscape manages to go awry. I can do neat things like play two chess engines against each other, each on a different processor—that makes for roughly 200,000 positions per second per processor. Compiling *anything* is a joke—it takes more time to type **tar -xzf**, **cd** and **./configure; make**. In fact, although fast machines are nice as servers, the amazing compilation speed would probably save much time and money for software development firms, not to mention improve the quality of software, as one can make far more trials in a given amount of time. (Well, I for one am a practicioner of the trial-and-error approach to programming.) Besides, it boosts morale to have fast computers.

Table 1 shows the results of some common Linux benchmarks, for those interested. Keep in mind, however, that the BYTEmark tests check only a single processor, so practically speaking, the results should be doubled.

## The Future

The future of VA Linux is a subject of optimistic speculation. VA seems to have adopted Sun's commitment to top-quality hardware with Dell's business model and is already the most famous exclusively Linux systems provider. After the Red Hat episode, it seems like everyone hopes to get in on the next Linux IPO, and rumors have been circulating about VA. Finances aside, the technology that will emerge from the organization formerly known as VA Research is bound to be impressive. Already some big names have gone to VA, the most prominent being Theodore T'so, Leonard Zubkoff, the Enlightenment team of Mandrake (Geoff Harrison) and Raster (Carsten Haitzler), Nettwerk (San Mehat), the nickname-less Michael Jennings, as well as Mark Vojkovich of XF86 fame, H.J. Lu of GNU/Bintools and NFS, and many others who deserve more mention than they receive.

What would you need brain power like that for? Try porting the Linux kernel to the new 64-bit Merced chip as part of the Trillion project, which is apparently going much smoother than Microsoft's port of Win32. VA is also developing VACM (a cluster manager to be released under the GPL), the Enlightenment

Window Manager, Perl bindings for GTK, the Linux kernel (in areas such as file systems, RAID, and large memory, Ethernet, and file system support), XFree86 and card drivers (Matrox in particular), and apparently some work on glibc.

## Conclusion

I have had the pleasure of using this machine for a couple months now, and it has done all sorts of things such as serving web-based e-mail, web-site hosting, code development, graphics programming, audio work and all the usual network things, as well as running all sorts of distributions. (You're not stuck with Red Hat, by the way, just copy the config files and install whichever distribution you like.) Any software I tested in the last few months was tested on the VAr. Modern software is made for slower, single-processor machines—everything runs better on the VArStation. Compilation times are short, graphic manipulation is instantaneous and almost never leaves tracers or flickers. The hardware stands up to everything. I even told the machine that it's the year 2000 right now, and everything still works—such a surprise! As far as improvements, I would like to see better audio support and digital flatscreen LCD monitors. Also, the keyboards feel nice, but have Windows keys and that's irritating. On a sillier note, the color scheme is fine and all, but I'd prefer something sinister like black. Nevertheless, whatever color it is and however many keys it has, it is still the fastest, most stable computer I have ever used.

**Jason Kroll** is fond of the post-apocalyptic terror of After Y2K! but he isn't really scared. In his dreams, vi will stop working and he'll be allowed to use Emacs. His computer related interests include artificial intelligence, parallel processing and microkernels, though he also likes music, art and chess quite a bit.

Archive Index Issue Table of Contents

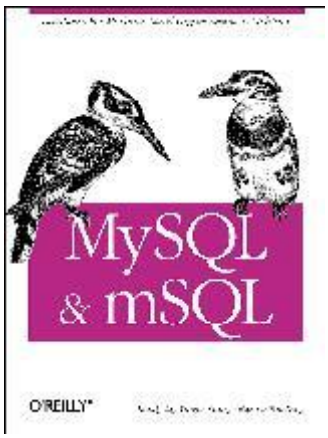
Advanced search

# MySQL & mSQL

**Reuven M. Lerner**

Issue #67, November 1999

This book has an excellent introduction to the theory behind relational databases and explains data normalization—a crucial aspect of good database design—in a relatively clear and concise way.




- Authors: Randy Jay Yarger, George Reese, and Tim King
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $34.95 US
- ISBN: 1-56592-434-7
- Reviewer: Reuven M. Lerner

Relational databases are based on technology originally invented in the 1970s. Their technology might be old, but databases play a crucial role in our increasingly information-laden world. They are designed to store and retrieve information easily, quickly and reliably.

With the growth of the World Wide Web, many more people have begun to realize how useful a relational database can be. Personalized web pages,

shopping carts, calendar/diary sites, stock portfolios and book recommendations are all possible because of the marriage between databases and the Web.

While creating and publishing a web site is relatively inexpensive, databases are normally expensive to buy, require costly hardware, and occupy one or more full-time database administrators. No wonder companies can charge outrageous sums of money for a database—anyone who can afford to install and maintain one can also afford the five-digit purchase price.

The tide is beginning to turn, however, and MySQL & mSQL are two at the forefront of a movement to popularize database use. They can be installed within minutes, require low maintenance, and have outstanding performance. Moreover, they support most of the standard SQL (Structured Query Language) syntax used on larger databases. This means anyone trained on a larger database system will be able to apply that knowledge with MySQL & mSQL. The reverse is also true; MySQL & mSQL allow someone to learn the basics of a relational database before working with Oracle, Sybase, DB2 and other large-scale products.

*MySQL & mSQL*, new from O'Reilly and Associates, is a good book for someone looking to use one or both of these products. It has an excellent introduction to the theory behind relational databases and explains data normalization—a crucial aspect of good database design—in a relatively clear and concise way. It describes SQL data types, giving frequent examples of how they could be used. I normally use MySQL, and thus, the mSQL portions of the book were irrelevant to my day-to-day work. Nevertheless, I found it useful to see how the two differ and what I might gain or lose by switching. Someone inexperienced with database programming would find it hard to decide between the two, based on the coverage given in this book.

*MySQL and mSQL* begins by describing databases in general, and then concentrates on these two database products in particular. The book continues with a look at database programming from a number of perspectives, including a wide variety of languages (C, C++, Java, Perl, Python and PHP). Finally, it provides a reference to SQL and several language-specific APIs.

That said, several things about *MySQL & mSQL* troubled me. I was surprised that the example CGI programs did not use the **-T** (taint) flag, which can significantly increase a program's security. I would also have preferred to see a longer description of Perl's DBI (database interface), instead of using space for a description of the old Mysql.pm module.

I am starting to learn Python and was happy to see that *MySQL & mSQL* included a chapter on the subject. Unfortunately, the book did not tell me where a MySQL module for Python could be found or which module was used in the examples. After an hour of searches, downloads and failed installs, I gave up. The authors could certainly have given clearer instructions on this subject.

I was surprised that the **DESCRIBE** command, which I use extensively in my work, had only one mention in the index—and that pointed to an empty entry in the book's SQL reference.

Beginning users probably need a short, step-by-step demonstration of how to **CREATE** a table, **INSERT** a row into it, **UPDATE** one of its columns, **DELETE** the row, and finally, **DROP** the table. *MySQL & mSQL* describes such commands in great detail, but gives few examples of interactive database commands.

Perhaps my biggest problem with *MySQL & mSQL* is its lack of attention to potential problems when using these low-end databases. I have been using MySQL for several years, and will continue to use it for many applications, both for myself and my clients. But it lacks useful items such as triggers, stored procedures and nested **SELECT** statements. More importantly, it lacks transactions and constraints, which help to ensure data safety.

Relational databases typically require much horsepower and maintenance, and the authors would have been wise to give a longer indication of why this is the case. In particular, it would have been nice to read more about transactions, and under what circumstances the added expense of Oracle or Sybase pales in comparison with corporate liability.

Despite its flaws, *MySQL & mSQL* is a welcome book. It provides far better documentation than has previously been available for these products, and gives a push in the right direction to programmers who are new to relational databases. If you are interested in writing applications that use a database, *MySQL & mSQL* is a good place to start.

**Reuven M. Lerner** is an Internet and Web consultant living in Haifa, Israel, who has been using the Web since early 1993. His book Core Perl will be published by Prentice-Hall in late 1999. Reuven can be reached at reuven@lerner.co.il.

# EMU—Event Management Utility

**Jarra Voleynik**

**Anna Voleynik**

Issue #67, November 1999

The authors present a freely available tool for monitoring enterprise systems through simplicity toward complexity.

Event management is a backbone of every enterprise management system, since it is the flow of events that describes various activities in the enterprise. By coordinating or acting on those events, enterprise starts being managed.

One of us, Jarra, has been contracting to a large computer company that was deploying Unicenter TNG for monitoring outsourced systems. Most of the systems were UNIX-based. Unicenter TNG agents, as is well known, are SNMP-based. After a considerable amount of time was spent tweaking the agents to come close to what he had decided to monitor and how to go about it, he arrived at the following conclusions:

- SNMP results in thick agents that are inflexible for dynamically changing environments.
- SNMP OIDs (Object IDentifiers) are not suitable for quick deployment of new resource hierarchies.
- Without access to the source code, which is typical of most commercial software, there is no way to extend the functionality of existing agents.
- The provided event management language is too restrictive and just another new language to learn.
- Linux is a superior management platform to Windows NT, especially in terms of available tools and utilities.
- The event console design determines the efficiency of the whole system.

We began thinking about how to accomplish system monitoring with ultra-thin clients, while maintaining their status at the event manager. That was when we

thought of using a principal parameter we coined “time-to-live”. At that very moment, EMU (Event Management Utility) was born. Each event message sent to EMU must have time-to-live set. It determines how long the message is kept in the database before it is deleted. If time-to-live is slightly more than the polling interval, the message will stay in the database until we stop resending it. In other words, the message will be in the database for the duration of an alarm only.

EMU is a comprehensive event management system with no limitations on what it can do for you. It harnesses the full power of Linux/UNIX utilities and scripting languages. Remember, software is here for us, and not the other way around.

### Event Management System Background

Event management systems usually consist of four components: monitoring manager, monitoring agents, event manager and event console.

### Monitoring Manager and Agents

Monitoring agents typically poll a resource, such as a file system, at regular intervals in order to track its status changes. These changes are communicated to the manager. The manager processes incoming resource status changes and either forwards them to the event manager or updates the business rules it maintains. The monitoring manager can poll for resource status information as well.

Resources may be of any kind, such as file systems, processes, disk, swap, applications or logged-in users.

Some monitoring agents, such as Unicenter TNG, are built around SNMP. In those systems, agents use traps to notify monitoring managers of status changes. Traps are sent using the UDP protocol, hence their delivery is not guaranteed. This drawback has to be made up for by regular polling by the manager, thereby collecting resource status independent of traps.

SNMP agents maintain status information of each resource they monitor. This information is typically maintained by the monitoring manager as well. Both agents and the manager keep resource status stored in a database. This is a key design feature of SNMP-based monitoring. Let us consider a file system example. If a threshold is set at 90%, an alarm is raised on exceeding this threshold. The status of the resource is changed to critical. It must stay in the critical state until the file system is cleaned up. Then the status is changed back to normal.

A need to maintain resource status and resource hierarchy results in rather complex SNMP agent coding. As a result, it is not trivial to extend existing agents. Furthermore, C coding skills and access to the source code are necessary.

### Event Manager and Console

The monitoring agent forwards events to the event manager, which usually performs some kind of event processing, such as matching messages against predefined strings and triggering an action in case of a match. Unicenter TNG provides a language for matching messages and taking action on them. The language is extremely simple, and thus inadequate for performing more complex correlations and actions.

The event console is the front end for administration staff. It allows us to “view” events. Its design determines its usability and efficiency. Events representing one alarm should be displayed as one event, rather than getting a separate visible event at each poll. To keep track of events easily, one strives to keep their display representations to a minimum. Otherwise, they scroll out of view before IT staff can take note of them.

### Problems with Enterprise Management Offerings

There are a few major players in the enterprise management arena, most notably Tivoli, BMC and Unicenter TNG. All of these strive to win support from other software and hardware vendors for their APIs. At the end of the day, they try hard to sell all the enterprise management components from their offerings. We find this disadvantageous to the end customer. Enterprise management is about using the best breed of products, from independent vendors, which can be easily integrated. This integration should be accomplished by either a single standard API or a command-line and script interface. Being UNIX types, weprefer a command-line interface and scripts. That way, we can intercept messages passed between enterprise management components. If a new product needs to be integrated, it can literally be plugged in.

### EMU Background

EMU is a flexible and integration-ready event management tool developed under Linux. It consists of a manager and agents. In fact, it integrates monitoring and event manager into one program. Agents are very simple scripts invoked by **cron**. These scripts are run at regular intervals, perhaps every five minutes. Each run scans the resources it monitors, comparing their thresholds against a configuration file. If a threshold is exceeded, a message is sent to the manager.

EMU employs time-to-live, which proved to be a simple way of maintaining resource status across agent polls. Let us suppose an agent that runs every five minutes found a resource problem. At each poll, it will send an alarm message to the manager. In this case, time-to-live will be set to slightly more than five minutes at 330 seconds. The manager will maintain the first message sent and its updates. If no update is received within the 330 seconds time-to-live of a particular message, the message is deleted and the problem is assumed to have been fixed. This simple approach allows us to write simple agents, preferably scripts, that scan a monitored resource and send their findings on each poll. The manager takes care of the rest. In fact, thanks to EMU, agents consisting of a few lines of code can monitor a very complex resource.

ASCII and Tcl/Tk interfaces to EMU are available. They represent a console for viewing events. The console displays all the necessary information to keep IT staff up to date. Each event is uniquely represented by a resource ID, which consists of the monitored system host name and object ID. All updates from the same resource ID are treated as one displayable event, while all the individual updates are stored in an event log file. If no updates for a resource ID have been received within an agent-specific time-to-live, the event message is removed from the console.

Examples of resource IDs are dumbo.company.com.au:/usr/local, tcc2345:sendmail and brk23:tz45. The first field before the colon designates a host name; the second field is a unique resource name. Two resources on a single system must not have the same resource ID, because EMU would treat them as the same resource.

The input interface to the manager is **emsg**, a small utility that uses TCP sockets to send messages to the manager. While the manager is written in Perl, emsg is written in C to facilitate its easy deployment on any monitored platform. In fact, Jarra is currently contracting to a company to install emsg on Linux, Compaq Tru64 UNIX, Solaris, AIX, IRIX, Sinix, Ultrix and VMS.

The integration interface is taken to the extreme by invoking input, delete and output scripts. Depending on the type of message, these scripts are issued on receipt of the message, on its removal or on its processing. All the message attributes are passed to the script as environment variables. In this way, we have achieved integration with Unicenter TNG event management. The TNG console-held area is, in fact, an exact image of the EMU console, thus making it much more usable and efficient.

EMU was built with distributed processing in mind. Multiple managers can run on a single or several systems, thus forming a hierarchy reflecting a company's

need. Through the truly open architecture of EMU, it is easy to synchronize multiple managers, build fail-over configurations or extend their functionality.

## Installation and Configuration

EMU consists of a manager (**gemu**), cleaner (**gemucleaner**), emsg agent (**emsg1**) and console/browser (**eb**, **xeb**). The manager and cleaner must run on the same node. The cleaner process manages message expirations. In order to provide flexibility, only one option is passed to gemu, gemucleaner and eb—the port number the particular server is running on. Both gemucleaner and eb use emsg to send delete messages to EMU.

A configuration file used by gemu, gemucleaner and eb is stored in /usr/local/emu/conf/*port#*.cfg. The configuration file describes the location of the EMU database (DBM-based), location of log files, scan interval for gemucleaner, etc. Each server will access its own configuration file based on the port number. If it suits your site, put the database under /usr/local/emu/*port#*/db. Each port/server will have log files and action scripts stored under /usr/local/emu/*port#* in sub-directories named logs and actions. The binaries/scripts are shared and stored in /usr/local/emu/bin.

One option in the configuration file is the location of emsg. If emsg, compiled for the individual platforms on your site, is stored in the /usr/local/emu/EMSG directory, you are ready to run eb (EMU browser/console) locally on your workstation. This is accomplished by exporting as read-only the /usr/local/emu directory. This directory will be mounted on the workstation as /usr/local/emu. By creating a symbolic link /usr/local/bin/emsg that points to /usr/local/emu/EMSG/emsg.*platform* and putting /usr/local/emu/bin in your search path, eb will run locally while displaying event messages from the server.

Depending on what actions EMU is configured to handle, the user ID it is running under can be either “emu” or “root”.

For the input, delete and output scripts, message attributes (e.g., host name, message text) are passed as environment variables. These can easily be used to trigger actions. It is a good idea to have one launcher script that, depending on message attributes, calls other, task-specific scripts. As a result, the workload imposed on the manager system will be reduced. The output script can be used to selectively forward messages to either a higher-level EMU or a third-party system. The input script may be used as a barrier to stop certain messages from processing based on a calendar. If this script returns a value greater than 0, the message is discarded. The delete message can be used for synchronization with a third-party system.

Time to live can be specified as seconds, minutes, hours or a fixed time in the form of HH:MM. A time-to-live of **-1** stands for infinity, and the associated message will be displayed in reverse video (by eb) and the cleaner will not expire it. The only way to put the message away is with a delete command on the console. This allows a batch job or backup failures to wait until they are acknowledged. Time-to-live set to **0** is used with so-called pass-through messages. They are not stored in the EMU database (they are recorded in the log file), but are intended to trigger an action.

Figure 1. ASCII-Based EMU Message Browser

The eb console provides a basic display of messages. A new message is displayed in bold to draw our attention. A message can be deleted/ acknowledged or annotated. Message annotations appear indented under each message. They serve the purpose of notifying others about details, such as a work request that has been logged. The message time shown on the console was the local time on the system that sent the first message. It helps identify when a problem occurred.

Figure 2. GUI-Based EMU Message Browser

EMU maintains a separate log file for each day. This log file stores all received messages, including their attributes, e.g. host name, message text and class. Message attributes are delimited with a vertical bar to allow for easy processing in scripts or uploading spreadsheets.

## Classes

The class option of emsg can be used in many powerful ways. When monitoring systems, it is best used for identifying a class hierarchy to the monitored resource; for example, /LINUX/PRO to designate a process subsystem or / LINUX/FS to designate a file system subsystem. In a way, it is similar to the SNMP OIDs; however, emsg class is much more flexible and can be created immediately as a need arises. Companies should develop a standard document detailing the classes format to be used. It is likely to reflect their business, resource and escalation hierarchies.

In a pure SNMP environment, a message arrives with an OID number that many people find cryptic and impractical. With the use of classes, the information is not only easy to read, but also lends itself to message filtering, forwarding, actioning, etc. For example, database messages may have a class set to /IT/ ORACLE. On receipt of such a message, the DBA may be paged to attend to the problem.

## Agents

In this section, a simple example agent for file system monitoring is demonstrated. Considerations are made of important aspects of the system along with tradeoffs. To make the example simple, the configuration file used will ignore minimal disk space limits for each file system. The code for this agent is shown in Listing 1.

Listing 1.

Once a resource is selected, we have to determine whether there is a periodicity in the way the resource can be monitored. For periodic monitoring, we need to know how frequently the resource should be monitored. The shorter the interval, the more resource-intensive the agent. However, by selecting too large an interval, we may miss an alarm in its early stage. For our file system agent, we will select a five-minute interval.

Next, time-to-live needs to be established. Given the poll interval of five minutes, we will select a time-to-live of six minutes. Remember, this must always be larger than the poll interval to keep display of events “continuous”. To achieve regular polling, the agent will be running from cron.

Once you have the agent, all that needs to be done is deciding which user it will run under and create a cron job for submissions in five-minute intervals. In fact, the simple code in Listing 1 is a full-blown agent for monitoring file systems with a 10% alarm limit.

## Actions

Now, let us put together a simple output action script. We are going to use EMU for monitoring a flow of events. To accomplish this, a directory called events is created. This directory stores files with names reflecting event names. If a file exists, it means the event it describes is active. Once the file is removed, the event has finished. Consider a scenario where a backup of SAP_ORACLE must complete by 6 AM. If a backup event file is found after 6 AM, it indicates the backup is running overtime or the backup script crashed without an opportunity to remove the file.

The SAP_ORACLE backup script reads as follows:

```
#!/usr/bin/ksh
emsg -n emuserver -p 2345 -t 0 -s 3 -w
icecream\
-c ADD_EVENT -m SAP_ORACLE_BACKUP
# start backup
.
.
.
# backup finished
```

```
emsg -n emuserver -p 2345 -t 0 -s 3 -w
icecream\
-c DEL_EVENT -m SAP_ORACLE_BACKUP
```

The output action script that creates or removes the event file will look as follows:

```
if [ "$E_CLASS" = "ADD_EVENT" ];then
        touch /usr/local/emu/events/$E_MSG
fi
if [ "$E_CLASS" = "DEL_EVENT" ];then
        rm /usr/local/emu/events/$E_MSG
fi
```

Another example is an input action script that stops messages from a node called dumbo, even though the EMU password is correct. It is necessary to mention an environment variable called **E_RHOST**. In order to facilitate forwarding of messages from EMU to EMU, emsg has an **-h** option for changing the name of the host from which the message arrived. This message attribute is stored in **E_HOST**. However, **E_RHOST** stores the true node name from which the message arrived. The input action script is as follows:

```
if [ "$E_RHOST" = "dumbo" ];then
        exit 1
else
        exit 0
fi
```

## Conclusion

Event management and resource monitoring is a complex subject, so we tried to touch on only the most important aspects of it. We believe by providing a free tool, enterprise event management will become a must on most sites. Linux is the best platform for EMU, since to take full advantage of its capabilities, an open and tools-rich environment is necessary. Check our web site at http://www.jarrix.com.au/ for the latest developments on the EMU front. Through collaboration around the globe, a valuable repository of EMU agents can be built. If you have an idea or have written an agent, let us know and we will post it on the EMU home page. If you have not done so yet, download EMU and delve into the vast and exciting horizons of enterprise management.

Resources

**Jarra Voleynik** has been involved with UNIX for the past 11 years. He is a graduate of the Technical University of Prague with an MS in Electronics. His first encounter with Linux two years ago got him hooked. He works as a UNIX consultant for Jarrix Systems. He can be reached at jarrix@yahoo.com.

**Anna Voleynik** (MS degree in Electronics) started being actively “aware” of Linux a year ago. She works as a UNIX Systems Administrator and keeps trying to minimize her and Jarra's “talking UNIX” at home. She mostly spends her spare

time with their children, ages 8 and 2. She can be reached at anna_vol@yahoo.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL



# Working with LWP

**Reuven M. Lerner**

Issue #67, November 1999

This month Mr. Lerner takes a look at the library for web programming and its associated modules.

Most of the time, this column discusses ways in which we can improve or customize the work done by web servers. Whether we are working with CGI programs or **mod_perl** modules, we are usually looking at things from the server's perspective.

This month, we will look at LWP, the “library for web programming” available for Perl, along with several associated modules. The programs we will write will be web clients, rather than web servers. Server-side programs receive HTTP requests and generate HTTP responses; our programs this month will generate the requests and wait for the responses generated by the server.

As we examine these modules, we will gain a better understanding of how HTTP works, as well as how to use the various modules of LWP to construct all sorts of programs that retrieve and sort information stored on the Web.

## Introduction to HTTP

HTTP, the “hypertext transfer protocol”, makes the Web possible. HTTP is one of many protocols used on the Internet and is considered a high-level protocol, alongside SMTP (the simple mail transfer protocol) and FTP (file transfer protocol). These are considered high-level protocols because they sit on a foundation of lower-level protocols that handle the more mundane aspects of networking. HTTP messages don't have to worry about handling dropped packets and routing, because TCP and IP take care of such things for it. If there is a problem, it will be taken care of at a lower level.

Dividing problems up in this way allows you to concentrate on the important issues, without being distracted by the minute details. If you had to think about

your car's internals every time you wanted to drive somewhere, you would quickly find yourself concentrating on too many things at once and unable to perform the task at hand. By the same token, HTTP and other high-level protocols can ignore the low-level details of how the network operates, and simply assume the connection between two computers will work as advertised.

HTTP operates on a client-server model, in which the computer making the request is known as the client, and the computer receiving the request and issuing a response is the server. In the world of HTTP, servers never speak before they are spoken to—and they always get the last word. This means a client's request can never depend on the server's response; a client interested in using a previous response to form a new request must open a new connection.

### Sending a Request

Given all of that theory, how does HTTP work in practice? You can experiment for yourself, using the simple **telnet** command. **telnet** is normally used to access another computer remotely, by typing:

```
telnet remotehost
```

That demonstrates the default behavior, in which telnet opens a connection to port 23, the standard port for such access. You can use telnet to connect to other ports as well, and if there is a server running there, you can even communicate with it.

Since HTTP servers typically run on port 80, I can connect to one with the command:

```
telnet www.lerner.co.il 80
```

I get the following response on my Linux box:

```
Trying 209.239.47.145...
Connected to www.lerner.co.il.
Escape character is '^]'.
```

Once we have established this connection, it is my turn to talk. I am the client in this context, which means I must issue a request before the server will issue any response. HTTP requests consist, at minimum, of a method, an object on which to apply that method, and an HTTP version number. For instance, we can retrieve the contents of the file at / by typing

```
GET / HTTP/1.0
```

This indicates we want the file at / to be returned to us, and that the highest-numbered version of HTTP we can handle is HTTP/1.0. If we were to indicate

that we support HTTP/1.1, an advanced server would respond in kind, allowing us to perform all sorts of nifty tricks.

If you pressed **return** after issuing the above command, you are probably still waiting to receive a response. That's because HTTP/1.0 introduced the idea of "request headers", additional pieces of information that a client can pass to a server as part of a request. These client headers can include cookies, language preferences, the previous URL this client visited (the "referer") and many other pieces of information.

Because we will stick with a simple **GET** request, we press **return** twice after our one-line command: once to end the first line of our request, and another to indicate we have nothing more to send. As with e-mail messages, a blank line separates the headers—information about the message—from the message itself.

### Examining the response

After typing **return** a second time, you should see the contents of http://www.lerner.co.il/ returned to you. Once the document has been transferred to your terminal, the connection is terminated. If you want to connect to the same server again, you may do so. However, you will have to issue a new connection and a new request.

Just as the client can send request headers before the request itself, the server can send response headers before the response. As in the case with request headers, there must be a blank line between the response headers and the body of the response.

Here are the headers I received after issuing the above GET request:

```
HTTP/1.1 200 OK
Date: Thu, 12 Aug 1999 19:36:44 GMT
Server: Apache/1.3.6 (UNIX) PHP/3.0.11 FrontPage/3.0.4.2 Rewrit/1.0a
Connection: close
Content-Type: text/html
```

The above lines are typical for a response.

The first line produces general information about the response, including an indication of what is yet to come. First, the server tells us it is capable of handling anything up to HTTP/1.1. If we ever want to send a request using HTTP/1.1, this server will allow it. After the HTTP version number comes a response code. This code can indicate a variety of possibilities, including whether everything went just fine (200), the file has moved permanently (301), the file was not found (404), or there was an error on the server side (501).

The numeric code is typically followed by a text message, which gives an indication of the meaning behind the numbers. Apache and other servers might allow us to customize the page displayed when an error occurs, but that customization does not extend to this error code, which is standard and fixed.

Following the error code comes the date on which the response was generated. This header is useful for proxies and caches, which can then store the date of a document along with its contents. The next time your browser tries to retrieve a file, it will compare the **Date:** header from the previous response, retrieving the new version only if the server's version is newer.

The server identifies itself in the **Server:** header. In this particular case, the server tells us not only that it is Apache 1.3.6 running under a form of UNIX (in this case, Linux), but also some modules that have been installed. My web-space provider has chosen to install PHP, FrontPage and Rewrit; as we have seen in previous months, mod_perl is another popular module for server-side programming, and one which advertises itself in this header.

As we have seen, an HTTP connection terminates after the server finishes sending its response. This can be extremely inefficient; consider a page of HTML that contains five **IMG** tags, indicating where images should be loaded. In order to download this page in its entirety, a web browser has to create six separate HTTP connections—one for the HTML and one for each of the images. To overcome this inefficiency, HTTP/1.1 allows for “persistent connections”, meaning that more than one document can be retrieved in a single HTTP transaction. This is signalled with the **Connection** header, which indicated it was ready to close the connection after a single transaction in the example above.

The final header in the above output is **Content-type**, well-known to CGI programmers. This header uses a MIME-style description to tell the browser what kind of content to expect. Should it expect HTML-formatted text (**text/html**)? Or a JPEG image (**image/jpeg**)? Or something that cannot be identified, which should be treated as binary data (**application/octet-stream**)? Without such a header, your browser will not know how to treat the data it receives, which is why servers often produce error messages when **Content-type** is missing.

HTTP/1.0 supports many methods other than GET, but the main ones are GET, **HEAD**, and **POST**. GET, as its name implies, allows us to retrieve the contents of a link. This is the most common method, and is behind most of the simple retrievals your web browser performs. HEAD is the same as GET, but quits after printing the response headers. Sending a request of

```
HEAD / HTTP/1.0
```

is a good way to test your web server and see if it is running correctly.

POST not only names a path on the server's computer, but also sends input in name,value pairs. (GET can also submit information in name,value pairs, but it is considered less desirable in most situations.) POST is usually invoked when a user clicks on the “submit” button in an HTML form.

### LWP::Simple

Now that we have an understanding of the basics behind HTTP, let's see how we can handle requests and responses using Perl. Luckily, LWP contains objects for nearly everything we might want to do, with code tested by many people.

If we simply want to retrieve a document using HTTP, we can do so with the **LWP::Simple** module. Here, for instance, is a simple Perl program that retrieves the root document from my web site:

```
#!/usr/bin/perl --w
use strict;
use diagnostics;
use LWP::Simple;
# Get the contents
my $content = get "http://www.lerner.co.il/";
# Print the contents
print $content, "\n";
```

In this particular case, the startup and diagnostics code is longer than the program. Importing **LWP::Simple** into our program automatically brings the **get** function with it, which takes a URL, retrieves its contents with GET, and returns the body of the response. In this example, we print that output to the screen.

Once the document's contents are stored in **$content**, we can treat it as a normal Perl scalar, albeit one containing a fair amount of text. At this point, we could search for interesting text, perform search-and-replace operations on **$content**, remove any parts we find offensive, or even translate parts into Pig Latin. As an example, the following variation of this simple program turns the contents around, reversing every line so that the final line becomes the first line and vice versa; and every character on every line so that the final character becomes the first and vice versa:

```
#!/usr/bin/perl -w
use strict;
use diagnostics;
use LWP::Simple;
# Get the contents
my $content = get "http://www.lerner.co.il/";
# Print the contents
print scalar reverse $content, "\n";
```

Note how we must put **reverse** in scalar context in order for it to do its job. Since **print** takes a list of arguments, we force scalar context with the **scalar** keyword.

### HTTP::Request and HTTP::Response

There are times, however, when we will want to create more sophisticated applications, which in turn require more sophisticated means of contacting the server. Doing this will require a number of different objects, each with its own task.

First and foremost, we will have to create an **HTTP::Request** object. This object, as you can guess from its name, handles everything having to do with an HTTP request. We can create it most easily by saying:

```
use HTTP::Request;
my $request = new HTTP::Request("GET",
   "http://www.lerner.co.il");
```

where the first argument indicates the request method we wish to use, and the second argument indicates the target URL.

Once we have created an HTTP request, we need to send it to the server. We do this with a "useragent" object, which acts as a go-between in this exchange. We have already looked at the **LWP::Simple** useragent in our example programs above.

Normally, a useragent takes an **HTTP::Request** object as an argument, and returns an **HTTP::Response** object. In other words, given **$request** as defined above, our next two steps would be the following:

```
my $ua = new LWP::UserAgent;
my $response = $ua->request($request);
```

After we have created an **HTTP::Response** and assigned it to **$response**, we can perform all sorts of tests and operations.

For starters, we probably want to know the response code we received as part of the response, to ensure that our request was successful. We can get the response code and the accompanying text message with the **code** and **message** methods:

```
my $response_code = $response->code;
my $response_message = $response->message;
```

If we then say:

```
print qq{Code: "$code"\n};
print qq{Message: "$message"\n};
```

we will get the output:

```
Code: "200"
Message: "OK"
```

This is well and good, but it presents a bit of a problem: How do we know how to react to different response codes? We know that 200 means everything was fine, but we must build up a table of values in order to know which response codes mean we can continue, and which mean the program should exit and signal an error.

The **is_success** method for **HTTP::Response** handles this for us. With it, we can easily check to see if our request went through and if we received a response:

```
if (!$response->is_success)
   {print "Success. \n";}
else
   {print "Error: " . $response->status_line . "\n";
}
```

The **status_line** method combines the output from code and message to produce a numeric response code and its printed description.

We can examine the response headers with the **headers** method. This returns an instance of **HTTP::Headers**, which offers many convenient methods that allow us to retrieve individual header values:

```
my $headers = $response->headers;
print "Content-type:", $headers->content_type,
   "\n";
print "Content—length:",
   $headers->content_length, "\n";
print "Date:", $headers->date, "\n
print "Server:", $headers->server, "\n";
```

### Working with the Content

Of course, the Web is not very useful without the contents of the documents we retrieve. **HTTP::Response** has only one method for retrieving the content of the response, unsurprisingly named **content**. We can thus say:

```
my $content = $response->content;
```

At this point, we are back to where we were with our **LWP::Simple** example earlier: We have the content of the document inside of **$content**, which stores the text as a normal string.

Listing 1

If we were interested in using **HTTP::Request** and **HTTP::Response** to reverse a document, we could do it as shown in Listing 1. If you were really interested in producing a program like this one, you would probably stick with **LWP::Simple** and use the get method described there. There is no point in weighing down your program, as well as adding all sorts of method calls, if the object is simply to retrieve the contents from a URL.

The advantage of using a more sophisticated user agent is the additional flexibility it offers. Whether that flexibility is worth the tradeoff in complexity will depend on your needs.

For example, many sites have a robots.txt in their root directory. Such a file tells "robots", or software-controlled web clients, which sections of the site should be considered out of bounds. These files are a matter of convention, but are a long-standing tradition which should be followed. Luckily, LWP includes the object **LWP::RobotUA**, which is a user agent that automatically checks the robots.txt file before retrieving a file from a web site. If a file is excluded by robots.txt, **LWP::RobotUA** will not retrieve it.

**LWP::RobotUA** also differs from **LWP::UserAgent** in its attempt to be gentle to servers, by sending only one request per minute. We can change this with the **delay** method, although doing so is advisable only when you are familiar with the site and its ability to handle a large number of automatically generated requests.

### Extracting Tags

Once we have retrieved the content from a web site, what can we do with it? As demonstrated above, we can print it out or play with the text. But many times, we want to analyze the tags in the document, picking out the images, the hyperlinks or even the headlines.

In order to do this, we could use regular expressions and **m//**, Perl's matching operator. But an easier way is to use **HTML::LinkExtor**, another object that is designed for this purpose. Once we create an instance of **HTML::Extor**, we can then use the **parse** method to retrieve each of the tags in it.

**HTML::LinkExtor** works differently from many modules you might have used before, in that it uses a "callback". In this case, a callback is a subroutine defined to take two arguments—a scalar containing the name of the tag and a hash containing the name,value pairs associated with that tag. The subroutine is invoked each time **HTML::LinkExtor** finds a tag.

For example, given the HTML

```
<input type="text" value="Reuven"
     name="first_name" size="5">
```

our callback would have to be prepared to handle a scalar of value **input**, with a hash that looks like

```
(type => "text", value => "Reuven",
     name => "first_name", size => "5")
```

Listing 2

If we are interested in printing the various HTML tags to the screen, we could write a simple callback that looks like Listing 2. How do we tell **HTML::LinkExtor** to invoke our callback subroutine each time it finds a match? The easiest way would be for us to hand callback to the parse method as an argument.

Perl allows us to pass subroutines and other blocks of code as if they were data by creating references to them. A reference looks and acts like a scalar, except that it can be turned into something else. Perl has scalar, array and hash references; subroutine references fit naturally into this picture as well. **HTML::LinkExtor** will dereference and use our subroutine as we have defined it.

We turn a subroutine into a subroutine reference by prefacing its name with **\&**. Perl 5 no longer requires that you put **&** before subroutine names, but it is required when you are passing a subroutine reference. The backslash tells Perl we want to turn the object in question into a reference. If **&callback** is defined as above, then we can print out all of the links in a document with the following:

```
my $parser = HTML::LinkExtor->new(\&callback);
$parser->parse($response->content);
```

Note that **$content** might have all HTML links that were returned with the HTTP response. However, that response undoubtedly contained some relative URLs, which will not be interpreted correctly out of context. How can we accurately view the link?

**HTML::LinkExtor** takes that into account, and allows us to pass two arguments to its constructor (**new**), rather than just one. The second argument, which is optional, is the URL from which we received this content. Passing this URL ensures all URLs we extract will be complete. We must include the line

```
use URI::URL;
```

in our application if we want to use this feature. We can then say

```
my $parser = HTML::LinkExtor->new(\&callback,
                       "http://www.lerner.co.il/");
    $parser->parse($response->content);
```

and our callback will be invoked for each tag, with a full, absolute URL even if the document contains a relative one.

Our version of **&callback** above prints out all links, not just hyperlinks. We can ignore all but "anchor" tags, which allow us to create hyperlinks by modifying **&callback** slightly, as shown in Listing 3.

Listing 3

### A Small Demonstration

With all of this under our belts, we will write an application (Listing 4) that follows links recursively until the program is stopped. This sort of program can be useful for checking links on your site or harvesting information from documents.

Listing 4

Our program, download-recursively.pl, starts at the URL called *$origin* and collects the URLs contained within it, placing them in the hash *%to_be_retrieved*. It then goes through each of those URLs one by one, collecting any hyperlinks that might be contained within them. Each time it retrieves a URL, download-recursively.pl places it in *%already_retrieved*. This ensures we will not download the same URL twice.

We create **$ua**, our instance of **LWP::RobotUA**, outside of the "while" loop. After all, our HTTP requests and responses will be changing with each loop iteration, but our user agent can remain the same throughout the life of the program.

We go through each URL in *%to_be_retrieved* in a seemingly random order, taking the first item returned by **keys**. It is obviously possible to sort the **keys** before taking the first element from the resulting list or to do a depth-first or breadth-first search through the list of URLs.

Inside the loop, the code is as we might expect: we create a new instance of **HTTP:Request** and pass it to **$ua**, receiving a new instance of **HTTP:Response** in return. Then we parse the response content with **HTML::LinkExtor**, putting each new URL in *%to_be_retrieved*, but only on the condition that it is not already a key in *%already_retrieved*.

You may find it interesting to let this program go for a while, following links from one of your favorite sites. The Web is all about linking; see who is linking to whom. You might be surprised by what you find.

## Conclusion

Our whirlwind tour of HTTP and LWP stops here. As you can see, there is not very much to learn about either of them; the trick is to use them to create interesting applications and avoid the pitfalls when working with them. LWP is a package I do not often need, but when I do need it, I find it indispensable.

Resources

**Reuven M. Lerner** is an Internet and Web consultant living in Haifa, Israel. His book Core Perl will be published by Prentice-Hall later this year. Reuven can be reached at reuven@lerner.co.il. The ATF home page is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

# Focus on Software

**David A. Bandel**

Issue #67, November 1999

Send Packet, adzapper, birthday and more.

By now, everyone knows what happened with the Microsoft challenge to the cracker community to break a Win2K box with IIS. The part I found the most hilarious was Microsoft's attempt to force these script kiddies to abide by some rules while they performed beta security testing. I guess Microsoft won't make that mistake again (at least not until Windows 2010). Do we need to take a look at some security software, and a program or two to help with the logs? From watching the LinuxPPC challenge, Linux looked like a state-of-the-art Fort Knox compared to Microsoft. Lest we get too cocky, remember that most distributions aren't this secure out of the box. LinuxPPC had most services turned off.

Send Packet:http://redirect.to/mg/

**Send Packet** will allow root to send TCP, UDP or ICMP packets on the network with any number of items set: source address, destination address, window size, TTL, ICMP code, etc. Wonder how your firewall will react to certain packets? Turn on logging and send a packet, then check the results. This should not be used by anyone except root for obvious reasons. It requires glibc.

adzapper:http://www.halcyon.com/adamf/adzapper/

Does anyone out there remember Slashdot without the banners? Would you like to go back? The **adzapper** is a small, configurable “proxy” that runs on your local machine (or any machine) to cover up those gaudy ads. Take back control of your content. I have been using this for a few days and will most likely keep running it. Sufficient “zaplets” (small files that define URLs to filter out) exist to clean up most sites, and you can always add your own. It requires Python.

birthday:http://user.bene-online.de/mschmid/

(Warning: this paragraph is not Y2K compliant; saving it on your computer system could result in your cows going dry and not giving milk—you have been warned.) Here is a simple little reminder program. Just create your ~/.birthday file as instructed by the documentation, put the **birthday** command in your startup file and get a message reminding you about upcoming birthdays. I suppose as long as your birthdays are Y2K compliant, the output will be, too. (Just don't complain to me if it doesn't remind you of anything on 31 Feb 00.) The only glitch I noticed was it didn't return the xterm back to its original foreground color, but left it white instead. It is difficult to read the text on a white xterm background. It requires glibc.

BASS:www.securityfocus.com/data/tools/network/bass-1.0.7.tar.gz

**BASS** (no home page) is a scanner to run from your local system. It has various features, including a "coward" mode. When invoked in this coward mode, it tries to hide by going to sleep when someone logs in. A normal process listing (**ps aux**) shows a blank in the "Command" column. However, a **ps auxc** or use of gps (see below) shows up the command just fine. Only root can perform certain functions, but it is unclear to me what any user other than root would be doing running this program. It requires glibc.

colortail:www.student.hk-r.se/~pt98jan/colortail.html

**colortail** allows you to view files with colors for designated text. The program comes with several configuration files for looking at log files, the very thing I had in mind when I downloaded it. It makes scanning log files much easier. The provided config files look much better on the black console background than on a white xterm background, but that problem can be easily fixed if necessary. It requires libstdc++, libm and glibc.

dhcpxd:http://www.dhcpxd.dhs.org:50080/

For those who use **dhcpcd** often, this client is great. **dhcpxd** has more features than dhcpcd, including support for aliasing. Several of my clients run dhcpcd, so I can just plug in my laptop, get an IP address for eth0:1, and get to work. You do need aliasing support compiled into the kernel, but that is small potatoes. It requires libstdc++, libm and glibc.

gps:http://www.dcc.unicamp.br/~guazzibe/gPS/

The **gps** package does what many **ps** packages do—shows you a list of processes. What makes this program a standout is that the author logically (in my opinion) arranged the output in an order more aligned to how people would use the columns: PID, command, owner, state, CPU%, Size, RSS, Nice,

Priority and date/time of Start. For me, this beats the standard layout where the two things I need most (PID and command) are on opposite sides of the table. Buttons for SIGHUP and SIGKILL (but not SIGTERM) are included. It lacks a way to specify columns and their display order. I look forward to grabbing a newer copy of this software soon. It requires libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libstdc++, libm and glibc.

qutar:http://home.sol.no/~geirerni/qutar/

**qutar** is a Qt interface to run tar and gzip, bzip2 or zip. It has a nice uncluttered feel. It is still early in development, but shows signs of being a well-thought-out program. About the only minor annoyance is when you do create or unpack a file, the directory windows don't update (yet). It requires libqt (v2.0), libstdc++, libm, glibc, libXext and libX11.

gomenu:http://www.digizen.net/members/geoffm/

Many years ago, I can remember creating .bat files for DOS systems so that techno-neanderthals could just type in a number and the program they wanted to run would start. Well, guess what? If you have a Linux system with too little RAM to efficiently run X, you can have exactly the same thing with this nice program. Talk about bringing back memories! These menus can be edited from the menu—very nice. Now, what did I do with that old 486-25 with 8MB RAM? It requires bash or pdksh.

**David A. Bandel** (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is co-author of Que Special Edition: Using Caldera OpenLinux, he plans to spend more time writing about Linux while relaxing and enjoying life in the “Crossroads of the World”.

Archive Index Issue Table of Contents

Advanced search

# Letters to the Editor

**Various**

Issue #67, November 1999

Readers sound off.

## Storm Clouds Brewing

As a regular reader from a far-flung shore, I felt compelled to write regarding a problem I see looming on the not-too-distant horizon.

The other night while I was trawling the Internet checking the availability of drivers for new hardware, I discovered something that gave me great cause for concern—the state of Linux support for the sleeping dragon that is USB.

Like most devices in the Linux world, support takes time, since dedicated coders often write drivers in their spare time. The speed of support for a given device is often directly related to the number of people needing that device. At present, Linux support for USB appears to be zero. This is not worrying in itself, but the development effort going into USB drivers also appears to be virtually zero, which is worrying.

Ironically spurred on by Apple's iMac and proper support in Windows 98, the USB has finally taken off big-time—witness the number of USB peripherals in the shops, with new ones being released almost daily. Again, this is not a particular problem, but soon manufacturers will be selling PC machines without serial or parallel ports, and probably without keyboard or mouse ports either, relying instead on the USB versions of such devices.

Apathy in the development of USB support may yet do what the combined might of Microsoft's strong-arm tactics has failed to do—relegate Linux back into the pack of “also-runs”, as people find they cannot use Linux on their increasingly standard machines.

—Nick Eller bysolo@cableol.co.uk

## IGEL

In the June issue interview with me, the web address for IGEL in Germany was misprinted. The correct addresses for our companies are:

Americas and Canada: http://www.igelusa.com/ and Infomatec IGEL Labs GmbH: http://www.igel.de/Infomatec AG: http://www.infomatec.com/

Thank you very much for your continued support. Keep up the good work!

—Hans L. Knobloch President & CEO, IGEL LLC

## Tech?

Is this a technical magazine or a sales pitch? Try putting in a few technical articles among the endless ads and Open Source flag waving.

Each issue is getting less and less informative. You are starting to look like *PC Magazine*. I had to ditch *Byte*, then *Sys Admin*. Is *Linux Journal* next? Is the magazine going to grow up or grow stale?

—Aaron C. Springer a.conrad@ix.netcom.com

It is true—the number of ads is rising along with the popularity of Linux. However, the proportion of content pages has remained about the same, because we have increased the size of the magazine to accommodate both more content and the ads. The October issue was 132 pages —Editor

## Wow!

I have been an avid reader of your publication for almost four years now, and you never fail to amaze me. This (August '99) issue of *LJ* blasts away all the other issues I have read. In fact, I just may re-subscribe (I hope the past problems with your fulfillment service have been resolved). What strikes me the most from this issue is the addition of “upFRONT”—very interesting and informative in a light fashion. Keep up the extraordinary work.

—David Comeau davitron@vl.videotron.ca

Thanks for the good words. As for the subscription fulfillment problems, we have taken subscriptions back in-house (announced in the October issue). Things should be working much smoother by the time this issue is out —Editor

### “Focus on Software” Column

You mentioned a package called “Ministry of Truth” that you reviewed in an earlier issue, which has since morphed into a job-tracking system. Could you give me a URL for the product? I might have a need for its new incarnation.

Thanks for the help, and thanks for the great column—it's my favorite!

—Chris Sherbak csherbak@tuc.com

It's at http://tomato.nvgc.vt.edu/~hroberts/mot/. The package I was talking about is the version 2 stuff. I've already used it to create a database of systems that I work on. Happy Linuxing.

—David A. Bandel david@pananix.com

### 1024 Cylinders

In response to a tech support question in the August issue of *Linux Journal*, Mario Bittencourt said:

> Using the installation's fdisk (or disk druid), create the partitions you will need with the first one for DOS/ Windows. Then separate a small (64-128MB) partition for swap and the rest for Linux. When you finish your installation, make sure you pick “MBR install” for LILO.

No dice; that won't prevent the problem. The kernel has to be in the first 1024 cylinders, and with your solution, it will be there only by coincidence. Build a new kernel on a very full drive, and it'll wind up outside the first 1024 cylinders —blammo, not bootable.

The other solution posted (creating a /boot partition in the first 1024 and putting your kernels there) is the best one I've managed to come up with. I usually go for 10MB instead of 8, but 8 is good too.

—Shawn McMahon smcmahon@eiv.com

Archive Index Issue Table of Contents

Advanced search

*More Letters*

## *Corel Linux Statement*

The talk continues about Corel's beta testing phase for Corel Linux and I wanted to send you our response to the criticisms that have arisen over the past few days. Hopefully this will clarify our beta testing phase and our commitment to the open source community. Please give me a call if you have any questions. Thanks.

A Time For Clarification

Corel wishes to clarify some issues that have arisen concerning the Beta testing phase of Corel LINUX.

The Beta testing phase for Corel LINUX, and the related Corel Beta Testing Agreement is in no way intended to contravene the terms of the GNU GPL or any other applicable Open Source license agreement. The restrictions on reproduction and distribution of the Beta version of Corel LINUX contained in the Beta Testing Agreement are intended to apply only to those components of Corel LINUX that were independently developed by Corel without the use of Open Source software.

Corel has decided to restrict the reproduction and distribution of these independently-developed components because further testing of these features is required before distributing them to a wider audience. As these added features will bear the Corel brand name, we wish to ensure that they are of the high quality that people expect from Corel.

In order to rectify some of the confusion surrounding the Beta testing program, Corel will be amending its Beta Testing Agreement to further delineate between Beta testers use of Open Source code components and Corel-developed code. As planned, beta testers will be receiving the source code to both the Open Source code and Corel modifications to them within the next week to 10 days, and are free to distribute these modifications in accordance with the terms of the applicable Open Source license agreement.

Corel will not be providing the source code to those portions of the Beta version of Corel LINUX that were independently developed by Corel without the use of Open Source software. As stated previously, once these features are deemed to be ready for general release, Corel will make the source code available under an Open Source license, the terms of which will be disclosed at a later date.

Corel remains committed to supporting the Open Source community, and has been working closely with the Linux community for the last few

years. Corel has also contributed a significant amount of work to the open source Wine Project, which allows 32-bit Windows applications to run on Linux, and has worked to make Linux more accessible to mainstream users via Corel LINUX.

Do you know about Corel's involvement with Linux? Visit us at http://linux.corel.com

—Judith O'Brien, juditho@corel.ca

---

***Article Suggestion***

You print, from time to time, a comparison of Linux distributions. You regularly print full reviews of commercial products for Linux (hardware, software, books, etc.). You print mini reviews in your 'Focus on Software' column which seem to be mainly development or connectivity tools; and your 'New Products' column which seem to be mainly commercial products.

The point I am trying to make is that apart from a few commercial desktop/productivity products you do not mention desktop software for Linux.

Your magazine, which I have been reading for at least 5 years, seems to cater almost exclusively to information systems and development people. I believe that it *must* address the desktop/productivity users as well.

Microsoft DOS/Windows became widely used because it started off as a desktop/productivity system. It only entered the information systems realm because enterprise managers wanted decided to save money by taking advantage of the already trained user base.

Many potential Linux users are already using some sort of MS or Apple system. They have a *lot* of both money (to purchase) and time (to learn) invested in productivity software. They may have a significant amount of money invested in games for themselves or their children. They may also have money invested in educational software for their children. Many would like to switch to a more powerful and stable system.

The thing that is holding them back is *cost*. Yes Linux is free (sort of) and they already have the machine. However, a machine with Linux loaded and running like a top doesn't help them do what they need done. They need software! Most of the people who want to switch to Linux can not afford the cash to replace all their software (e.g. productivity, games, educational). Even if they can afford the cash, many of them have been using DOS/Windows or Apple/Macintosh applications for a long time and therefore have a large numbers of data files which is more valuable than the cash cost of switching. They can not or do not want to incur the *cost* (in terms of time) to re-create their data files. Unless they can get

software which; 1, is *very* inexpensive; 2, can load their data files; 3, can export data files which others who have not switched can read, they simply will *not* switch.

The main idea is that they want to keep cost, both monetary and effort/ time, of switching as low as possible. Once they have switched, slowly over time, they would be willing to spend money on software, again.

—Milton Hubsher, milton@CAM.ORG

---

***To Mock a Pronouncing Bird: Re Linux Journal article, September 1999, p.14***

Hey, kids, let's change some of the C language! Yeah, sure, let's just decide to replace some words with our own words ... what?? What do you mean we can't do that?

I'm sure it's obvious to everyone that a few people can't simply change a portion of a widely agreed language or standard. What stuns me is how the computer community—so savvy about its own standards—widely ignores the existing standards of English grammar and pronunciation. Your September article surveying pronunciations of Linux recorded on the Web, while an interesting survey of contemporary pronunciation, demonstrates this common ignorance. But it did contain one valuable revelation.

English pronunciation has rules, just as any computer language does. Although imperfect, they are necessary to help avoid the language descending into chaos, and to help readers know how to pronounce words. In Standard English pronunciation the second vowel “u” makes the first vowel “i” long. Thus in the word “line” we do not say “linn-ee.” (Indeed, in this example, the “n” is doubled to indicate that the vowel is a short “i” pronounced “ih,” so if we were discussing “Linnux” the first vowel would be short.)

The article's revelation that Torvalds was named after the American scientist Linus Pauling puts the final nail in the coffin for the “Linn-ucks” camp. Linus has always been pronounced “Line-us” because the name was absorbed into English, and became pronounced according to the rules of standard English pronunciation. No one ever called Dr. Pauling “Linn-us.” Even in its original language, the name was NEVER pronounced with a short “i”, as “Lih.”

Readers should note that Linus Torvalds, Maddog and several others from the early days adhere to their own pronunciation which I have not seen noted elsewhere. As clearly heard at LinuxWorld during their speeches, they actually pronounce Linux as “Linnix,” not “Linnux.” This may result from the OS's origin in the OS Minix and the “ix” in Unix. (Note that we do not say “Oo-nix.”)

The manner in which this debate has proceeded is significant. At LinuxWorld, vendors were heard loudly mocking attendees who

innocently used their English language background and correctly pronounced the OS as “Line-ucks.” Along with the author of your article, many seem to feel that pronunciation is simply “received,” meaning that the way they hear many individuals pronounce it makes it right. This is obviously not true. Although English is a living language, we are not freed from the duty of occasionally weeding the verbal garden.

A worse example is the common mispronunciation of the word “router” as “raowter.” The word comes from the French phase “en route,” pronounced “on root.” Until the mid-1980s, the standard American and English pronunciation of this word was “root,” as evidenced by the pronunciation of the popular US television show “Route 66.” When Local Area Networks became common technology, introduced by companies populated with engineers from the American South and West, the origin of the word and its correct pronunciation were displaced by the Southwestern regional pronunciation, which has now infested even television shows such as Star Trek Voyager (“Re-raowt power to the shields!”). To adopt this pronunciation is to deliberately bury the history of the phrase. (Clearly the show's producers do not realize how bizarre this pronunciation sounds outside the American Southwest.)

It is understandable that many individuals with a strong engineering and computer background are weak in English, just as many with a background in languages are weak in mathematics. One can sympathize with their plight since standard English pronunciation is infrequently or poorly taught these days. But it is impossible to sympathize with the authors of many key documents in our industry such as the foundation documents these documents are full of spelling and grammatical errors, yet each came from academic or commercial organizations large and sufficiently prosperous to afford editing and proofreading services. Had they understood the errors of their linguistic ways, these scientists would surely have behaved differently.

So this is how the debate has proceeded—without background, = investigation or scholarship. Your article concludes that the web survey of pronunciations is “inconclusive” and that the pronunciation “Linn-ucks” appears to be the winner. Winner? Correct pronunciation is no contest. At least those in the industry responsible for using the English language to discuss the languages of computers should accept their responsibility to make this clear.

It's “Line-ucks,” and you've helped prove it. :-)

—Malcolm Dean, malcolmdean@earthlink.net

---

***LJ Photos***

I've been a *Linux Journal* subscriber for a little under a year now, and first of all let me say that I always love the magazine, it's one of a very few I will consistently read cover-to-cover, and the design is great; attention-grabbing covers, a clean and easy-to-navigate look all throughout the articles. But I must complain about the photography. I've been a

photographer longer than a Linux user, and it seems like a majority of the photos I see alongside the copy are nearly lifeless - the colors are unsaturated and the tonal range is flat. What prompted me to write this comment, though, was the cover of this month's issue - the figure is distractingly lit and the right side of his face is completely enshadowed. The magazine deserves something better to mirror what's inside, and it needs something more professional if it is to compete with Cindy Crawford (or, perhaps, Lara Croft) beside it on the newsstand.

I realize that my own experience no doubt exaggerates what I notice, but in comparison to the excellent design surrounding the photography, I figured that someone needed to call it to the editorial staff's attention. Perhaps *LJ* needs to consider adding a dedicated photo editor to the staff. I for one, though I love the magazine the way it is, would love it exponentially more if the portraiture and incidental shots had the same crisp, clean look of the graphics and the colorful glow of the screen shots they accompany.

Please keep up the great work!

Regards, Nate
—Nathan P. Willis, npw@acm.acu.edu

---

*Coming of Age*

I agree with Mark H Runnels (*LJ* Sept 99, page 6, Letters) that if we want to get Linux popular with the masses, a distribution is needed that converts the computer into a Linux Appliance. I hope that Red Hat, Caldera, and the other distributions that try to achieve this will succeed soon.

I do not agree with Mr. Runnels that this will be Coming of Age. An appliance produce the same result independent of the knowledge of the user. There is another use of computers as an enhancement to your brain, where the result depends on the knowledge of the user. This is a very important use of computers.

Consider the new chip that finds the DNA of a sample. A person that is not able to spell DNA, using an appliance with this chip will obtain the DNA of any number of samples. Such person will not be able to correlate these DNAs to other characteristics of the samples. A person with knowledge of DNA, correlations, computers, etc. is needed to put such information onto a computer and find why one sample is from a person susceptible to flu, while another is from somebody immune. *Linux Journal* is full of articles of this type.

For this reason, I hope that Slackware and the other distributions that let you free will prosper and continue to give us that choice. Notice that with the death of DOS in all its mutations, Linux is the only system that permits you to do with a computer what you want, the way you want. This is why I say “Linux makes you free.”

Dr. Michael E. Valdez, mikevald@netins.net

---

*Coming of Age*

I largely agree with most of Mark Runnels' letter - I still run windows myself, mainly because that's what we get given at work and I'm familiar with it. But if I had to recommend a PC to my mother, it would be an Apple iMac not a Wintel machine, because it comes closest to computing's holy grail “plug it in - it works”. I've just spent the best part of a week's spare time trying to get a new Wintel machine at home to work, so how any total beginner manages I don't know. Linux isn't yet for beginners, but it's robust enough so that it could be if suitably packaged. A “plug it in - it works” Linux distribution is really what is needed to get the system into the mass market.

—Edward Barrow edward@cla.co.uk or edward@plato32.demon.co.uk

---

*LJ Sept 1999*

I have been getting *LJ* for quite a while now and look forward to every issue. Of course I never send a message when I am happy, just when I am not, so take this in the context of a usually very happy customer.

In the “Cooking with Linux” section in the Sept 1999 issue you state that the articles in the section are all about “fun things we can do with Linux”. I was very eager to read the article on Multilink PPP because it is something I have been wanting to get working for Linux for ages, without success. However I was disappointed when the article basically had nothing to do with Linux, in fact as far as I can remember it did not even mention it. There were no followup references to the Internet where I could download drivers or pointers to web sites with more information on the use of MLPPP with Linux.

It was a good article but I don't know why it was included. If I want to read about different network protocols, which is an interesting topic, I can read general computer magazines etc. I hope the *LJ* keeps its Linux focus and does not start to import irrelevant articles.

Well there you have it, I feel better now. Its the only thing I have been unhappy about in all my issues so you are doing something right. Also thanks for the T-Shirt which I got when I went to Linux World Expo (I came all the way from Australia). Out of all the T-shirts I got the *LJ* one with “Geek by nature - Linux by choice” is the one I like to wear the most.

Cheers

—Rodos, rodney@haywood.org

---

*Linux in the Future*

I have to say that reading the letter from Mark Runnels in issue 65 made me feel that at least I was not alone in my struggles.I have been trying

for some months now to switch over from Windows 98 to Linux. I will say that now that I have finally succeeded in getting a Linux system working on my computer thanks to Mandrake and a 16 bit sound card I removed from an old 486 computer. I am now struggling to get the software I need installed let alone working. So far I have given up on Siag , just would not compile at all, Star Office due to the huge download required and no guarantee that it would install after that and I have finally given up on KDE Office due to the fact that it requires qt 2.0 which after hours and hours of trying I just have not been able to install due to apparent syntax errors during the 'make'. I for one would be only to happy to buy a copy of an office system in a box that I could install without having to compile and hunt for missing files etc.

Linux in my opinion is an excellent system I have found that the Internet facilities are second to none and being able to use KDM for file transfer has been unbelievably reliable. I have also only had minor system failures which could have well been my fault but unlike Windows the recovery was painless. I feel now that it is time for the software writers to decide are you doing this as a pure academic exercise or do the most successful way of installing software on Linux so how about you all come to an agreement that this is not a bad system and supply all software this way otherwise I really cant see a future for Linux as a system for the masses. While I consider myself as a newbie with Linux I have been involved with computers for 25 years now having done a stint as a programmer myself ,so I feel that if I am having problems there must be others who have tried and probably given up.

The final request I would like to make is to hardware manufacturers, would it really be so difficult or expensive to write drivers for Linux then we would be able to use all your fancy hardware and we would look forward to new devices being released instead of having to use older hardware just to get it to work.

—Stephen Chilcott, f.chilcott@xtra.co.nz

---

*Your column "Is KDE the Answer?"*
Dear Phil,

I read your column "Is KDE the Answer?" in *LJ* 66 (Oct. 99) and was disappointed to find some major inaccuracies in it which I would not have expected in a publication that is so close to the Linux community.

You write:

> Most distributions have adopted KDE as their default and, as far as I know, all distributions include KDE.

That is not correct. Debian currently does not include it in Debian GNU/Linux.

Debian proper (“main”) consists of 100% free software, as per the Debian Social Contract (http://www.debian.org/social_contract). In addition to “main”, we have “non-free” and “contrib” sections on our FTP servers; “contrib” consisting of free software that is dependent on non-free software (like KDE was with Qt 1).

At one time KDE was available from the Debian FTP sites in the “contrib” section. Discussion of the licensing issues regarding KDE resulted in its removal from the Debian FTP sites in October 1998. This was reported in a Debian news article (http://www.debian.org/News/1998/19981008) which also contains the analysis of the issue.

> The glitch in KDE development was the non-open licensing of Qt.

That is only part of the issue. If it were just this, Debian could provide KDE binaries in the “contrib” section.

The real problem, which still exists with Qt 2.0, is in the interaction between KDE's license (the GPL) and Qt's, both the non-free Qt1 license and Qt 2's QPL.

For details on it, please see the news item. The core of it is that the GPL imposes requirements on the licensing of libraries a GPL-ed application depends on (with an exception for major system components like e.g. the C library on a proprietary Unix); Qt's license isn't compatible with these requirements.

The GPL doesn't impose restrictions on use, thus it is perfectly fine to download KDE source and compile it on one's own system. It does impose restrictions on redistribution of binaries, which are a problem in the KDE case. The Debian project honours software licenses, and has honoured KDE's GPL by ceasing to redistribute KDE binaries.

> [...] With Qt 2.0, it is a non-issue. The Qt people saw the problem and created a new license that addresses it.

It addresses only Qt's freedom issue. The QPL (http://www.troll.no/qpl/) under which Qt 2 is released is indeed a free license according to the Debian Free Software Guidelines (http://www.debian.org/social_contract#guidelines), for which Troll Tech is to be applauded. Debian's Joseph Carter <knghtbrd@debian.org> has provided extensive input during the development of the QPL which has made it a better license from the free software perspective. Joseph also explained the GPL interaction issue to Troll Tech. Regrettably, Troll Tech have chosen not to make the QPL a GPL-compatible license. Thus, although Qt is now free, Debian still can't redistribute KDE binaries.

Reportedly, the KDE project have now recognised the issue and are switching away from Qt 1 to Qt 2 and from the GPL to a different, Artistic-like, license that does not have bad interaction with the QPL. Once this is done, KDE can be included in Debian proper (“main”).

> Most distributions have adopted KDE as their default

Debian is about free software and choice. We don't really have a default window manager, and I suspect that in similar vein we won't have a default desktop environment.

In the meantime, KDE packages in .deb format have been made available outside the Debian project, both by KDE and by the KDE Linux Packaging Project (http://kde.tdyc.com).

I'd appreciate it if you could correct these inaccuracies in a follow-up.

Regards,

—Ray, J.H.M. Dassen, jdassen@wi.LeidenUniv.nl

---

### *“Floppy Formatting” Letter in Oct '99 issue of LJ.*

Read that letter with interest and was surprised your editorial staff did not catch the last suggestion of writer, namely:

```
chmod 777 /usr/bin/fdformat
reboot
```

reboot? Reboot? What for?
Glenn G. D'mello, glenn@zaphod.dmello.org

---

### *Film Stuff*

I just got my latest issue of *Linux Journal*, and I was very interested in the story on the Houdini port. I hope you'll continue to run stuff like that.

It seems to me that one of the big Linux stories that you guys are covering pretty much on your own is the growth of Linux in the film industry. I remember reading about the Alpha farm they used to render Titanic in your magazine as well.

I'd really like to see a broader view of the topic, an article that seeks out people in the film industry who are using Linux, and tries to take the measure of the trend. Something that would talk to the Alias and Houdini people and the guy who worked on the Alpha, and whoever else is working with linux. Why are they using it? How do the ports work? When you use Linux farms to render frames in movies, do you write the code from scratch? Who does that stuff? Did they have trouble selling it to their bosses? Do they use Linux at ILM?

I know you're looking for writers, but even if you're interested in it, I wouldn't know how to proceed, so I don't know if it's realistic. I don't know anyone in the film industry, and I wouldn't know who to call. But if you're interested and can put me in touch with people who could open some doors, I'd love to take a shot at it.

In any event, I hope you'll keep up the excellent work. I enjoy your magazine very much.

—alex, alex@crawfish.suba.com

---

### *Letters*

I have just received *LJ* No. 66 (Oct 99) and read Mr Hughes's article about KDE. I was not pleased with what I read. Specifically he has the arrogance to presume that only the GNOME programmers can fix KDE and secondly that it is a bug that KDE defaults to a European paper size.

The last point was particularly irksome as for years we Europeans (can someone who is British say that?) have had to put up with incorrect spellings and the wrong date format, to name but two “bugs”.

It is just this kind of americocentric view that has lead me to decide not to renew my subscription to *LJ*.

—AFL, A.F.Lack@city.ac.uk

---

### *Copy Editing*

It's ironic that Harvey Friedman (Oct. *LJ*, pp. 116-117) complained that “maddog” Hall needs a better copy editor, when someone at your establishment allowed him to say “... after first determining whether the computer is comprised of hardware on which Linux can run.”

This mis-use of “comprise” (when “compose” is meant) is a solecism that I find particularly irritating. If you have a list of troublesome words and phrases that you grep for in each incoming manuscript, please add “comprise” to it.

—Andrew Young, aty@sciences.sdsu.edu

---

### *About “Is KDE the Answer?”*

Dear Mr Hughes,

I am probably about the 999th person to protest to you about your editorial “Is KDE the Answer?” in the most recent *LJ*. But I hope I can get your ear nonetheless.

You start the editorial with two questions: “Is looking like MS Windows good or bad?” and “Is KDE a better answer than GNOME?”

You spend half the editorial arriving at an answer to the first question with which I agree: “It depends.” For certainly some people, fresh in out of the wilderness, will like to have a Windows-like interface, for a while anyway. Whereas others will want to put childhood behind them. Fortunately everyone can have what suits them.

But of course it is the second question that is bothersome. It reminds me of a similar question: “Is emacs better than vi?” Haven't we all learned that this kind of question is silly and irritating?

Don't you value competition? don't you see the benefits that ensue? to name just one example: would Troll Technologies have modified their license for qt if they, and the KDE community, hadn't felt GNOME's breath at their backs?

To try to pick a “winner” between KDE and GNOME is like trying to pick a winner between emacs and vi. It is IMHO silly and, worse, unproductive. Let's have (at least) two wonderful editors, and let's have (at least) two wonderful windowing environments. That is the way for Linux and Open Source to make progress. That is what will benefit us consumers the most.

For the record, I am a charter subscriber to *LJ*, which I love, and: my window manager of choice is fvwm.

Best wishes,

—Alan McConnell, alan17@wizard.net

---

### *A Little Disappointed*

Although you publish very nice articles about Linux related issues, I must say that I am a bit disappointed by your journal lately. Over the years the number of articles about free software such as fancy graphical tools, libraries, interpreters, compilers, and articles about Linux' internals steadily declined, and the number of articles about commercial applications on Linux steadily increased. In the last issue I had to search for information between the advertisements and product reviews. It may be good business for your journal but I am not interested in this commercial stuff and I do not want to pay for reading about it. I will start looking for another source of information.

Sincerely,
—Fulko van Westrenen F.C.v.Westrenen@tue.nl

---

### *Letter re. October 1999 Issue*

I read with interest the article about 'dungeon games', more commonly called 'roguelikes'. While the article is fairly accurate for a one-page potted summary, it seems odd that it doesn't mention that ADOM - unlike the other games - is not free software; the source code is not even

available for inspection, let alone modification. *Linux Journal* is, after all, a magazine about a free software operating system.

In my teens, I was lucky enough to use a Sun 3/60 with a copy of Hack on; not just a compiled binary, but also the source code. Not only was there this fascinating game to play, but you could change the way it worked, provided you could work with this thing called UNIX. I was hooked; and (while clearly it's his choice) I think Thomas Biskup is doing the roguelike community (especially those who use platforms which ADOM has not been ported to) a grave dis-service by keeping his source code closed.

I'd also like to take issue with Phil Hughes' call to support KDE. There are two very real problems with having to pay to develop proprietary applications that use Qt. One is a moral one; if KDE becomes the de facto GNU/Linux desktop, people will have to pay Troll Tech when what they want is to develop for GNU/Linux (but not necessarily Qt) - and I don't see why Troll should get that money when it is actually being paid in order to work with an OS which is almost all the work of people other than Troll. The second is a more practical one; GNU/Linux will be more of a success if we preserve the current situation where it costs nothing to develop for it. I have no problem with proprietary applications competing fairly - if Corel or Applix can write such a good wordprocessor that I would rather pay $50 for it than get StarOffice or AbiWord for free, good for them - and a charge of $1,000 per developer is not the way to attract those applications.

—David Damerell, damerell@chiark.greenend.org.uk

---

***KDE vs Gnome***

I certainly don't want to get into the middle of declaring that KDE is better or worse than Gnome, particularly since I prefer WindowMaker over either one. The truth is, WindowMaker, in my opinion, is the least of the evils.

I'm probably older than Phil Hughes, and with over a million lines of C and assembler behind me, I'm not an easy convert to point and click, however, my observations may be of value.

1. We shouldn't endorse any particular program for the sake of providing a unified look for novices. If they want mediocrity instead of programs that have evolved because of their usefulness, then there's another OS which has plenty of what they're looking for.

2. Incorporating `applications' inside of window managers is the kind of monolithic failure that is too evident in the `other' OS. That is, if I have to run a specific window manager to use a file manager, I'll probably do what I do now - command line it or write a script.

3. Focusing on glitz, with its code bloat and execution crawl, may catch a few eyes, but it will never replace solid functionality - anyone remember the early versions of LapLink? It looked like the text version it was, but worked with great ease and lightning fast.

4. A window manager that requires one to learn how it works, instead of providing a learning experience about how computers, programmers, and users interact is like giving people food instead of teaching them to grow their own.

So is it a good idea to all jump on KDE? I think not, even if a clear case could be made for its immediate superiority. I think that we need to stay the course—Linux isn't just about presentation with some directed and lasting appearance. Linux is an ongoing evolution. Each can choose to use those parts of it which are consistent with one's ability and experience.

In the meantime, I'll probably keep using a clutter of xterms until a `real' window manager appears—nxterm -T `pwd` &.

Sincerely,

David Smead, smead@amplepower.com

---

*Comments on October 1999 Issue*

A few comments on a few of the articles in the October issue:

The KDE editorial: Very well written, especially as many of us can attest to the concept of “more sophisticated idiots”. However, I disagree with the point that seems to be made that the KDE and GNOME groups should pool their efforts to create a unified environment. Although a dedicated KDE user, I like linux for the choice it offers me. I can choose KDE or GNOME, depending on which I prefer. I usually introduce newbies to KDE first, as I feel it does make the transition from Microsoft a bit easier.

The article on writing CD-ROMS. I wonder why there was no mention of the -J option when mkisofs. The Joliet extensions to ISO9660 are an alternative to the Rock Ridge extensions (the -r) option. Although they were a Microsoft creation, they are still important based on how the created cd will be used. If the only purpose is backing up for a linux machine, Rock Ridge is better... however, if the CD will be shared on multiple machines, Joliet should be closely looked at.

However, excellent issue... I usually find at least a few articles in each issue that interest me (and I read all of them anyhow), but this issue is so far the best I have received.

—Jason Ferguson, jferguson3@home.com

---

### *KDE and GNOME*

I'm just a home office desktop Linux user. I use Linux in my work, to maintain my church's Web site, and for other personal stuff (including playing games). I'm not a programming wizard and don't make my living in the world of bytes and cyberspace. But I really love Linux and have been using it since I installed Slackware back when the kernel version was around 1.0.6. (Winter of 1994-95, I *think* it was.)

But I don't get it. What is the big push to settle on one desktop?

Frankly, I'm not to worried about it because anyone who's ever used Enlightenment (and, from what I've been able to glean, there are lots of us who have) is never going to be happy with KDE, but I do not like being told by the “powers that be” (including the publisher of my favorite Linux rag) that I'm some sort of turncoat for not understanding the absolute necessity for the continued health of the Linux movement of shutting down all GNOME development this very afternoon.

I've used KDE a lot. I prefer (and use) many of the KDE applications, but, as a desktop, it's mostly boring. And anyway, if I wanted “powers that be” to make such decisions for me, I'd be running some Microsoft offering.

—Maury Merkin, merkin@his.com

---

### *Chez Marcel*

Chez Marcel's article (*Linux Journal*, Sept. 1999) on French cooking inspired me to share some recipes of my own. The cooking metaphor is not new to computing, as Donald Knuth, in his forward to “Fundamental Algorithms” confesses he almost used “Recipes for the Computer” as its title. Without stirring the metaphor to vigourously, Chez' article gives me the opportunity to share two items of interest and give them the needed cooking flavor.

For some time, I've been concerned about what I regard are overuse or misuse of two programming constructs:

- temporary variables and files, and
- nested logic and decisions.

To continue the cooking analogy, these two may be thought of respectively as inconsistent or lumpy sauce, and uneven temperature. Realizing that we chef's like to work on one anohter's recipes, lets see what happens when we apply them to Chez Marcel's recipe, “Check User Mail”.
Before I'd read Marcel's article, my style of progamming used the tool metaphor. While not much of a chef, I now prefer the cooking metaphor, as it connotes more of a learning, and sharing model, which is what we do in programming.

Marcel's recipe is an excellent starting point for my school of cooking, as his receipe is complete, all by itself, and offers the opportunity to visit each of the points once. First, here is a copy of his recipe, without the comment header.

```
for user_name in 'cat /usr/local/etc/mail_notify'
do
   no_messages='frm $user_name |
      grep -v "Mail System Internal Data" |
      wc -l'
   if [ "$no_messages" -gt "0" ]
   then
      echo "You have $no_messages e-mail message(s) waiting." > /tmp/$
      echo " " >> /tmp/$user_name.msg
      echo "Please start your e-mail client to collect mail." >> /tmp/
      /usr/bin/smbclient -M $user_name < /tmp/$user_name.msg
   fi
done
```

This script isn't hard to maintain or understand, but I think the chef's in the audience will profit from the seasonings I offer here.
A by-product of my cooking school is lots of short functions. There are those who are skeptical about adopting this approach. Let's suspend belief for just a moment as we go through the method. I'll introduce my seasonings one at a time, and then put Chez Marcel's recipe back together at the end. Then you may judge the sauce.

One of the languages in my schooling was Pascal, which if you recall puts the main procedure last. So, I've learned to read scipts backwards, as that's usually where the action is anyway. In Chez Marcel's script, we come to the point in the last line, where he sends the message to each client. (I don't know samba, but I assume this will make a suitable function):

```
    function to_samba { /usr/bin/smbclient -M $1; }
```

This presumes samba reads from its standard input without another flag or argument. It's used: “to_samba {user_name}”, reading the standard input, writing to the samba client.
And, what are we going to send the user, but a message indicating they have new mail. That function looks like this:

```
    function you_have_mail {
       echo "You have $1 e-mail message(s) waiting."
       echo " "
       echo "Please start your e-mail client to collect mail."
    }
```

and it is used: you_have_mail {num_messages}. writing the message on the standard output.

At this point, you've noticed a couple of things. The file names and the redirection of output and input are missing. We'll use them if we need them. But let me give you a little hint: we won't. Unix(linux) was designed with the principle that recipes are best made from simple ingredients. Temporary files are OK, but Linux has other means to reduce your reliance on them. Introducing temporary files does a few things:

- leaves you with extra cleaning to do after the meal is served,
- forces you to make sure someone else isn't using the bowl when you need it for your recipe.

Therefore, we seek to save ourselves these tasks. We'll see how this happens in a bit.
A key piece of the recipe is deciding whether or not our user needs to be alerted to incoming mail. Let's take care of that now:

```
    function num_msg { frm $1 | but_not "Mail System Internal Data" | l
```

This is almost identical with Marcel's code fragment. We'll deal with the differences later. The curious among you have already guessed. This function is used: num_msg {user_name}, returning a count of the number of lines.
What does the final recipe look like. All of Chez Marcel's recipe is wrapped up in this one line of shell program:

```
    foreach user_notify  'cat /usr/etc/local/mail_notify'
```

And that's exactly how it's used. This single line is the entire program, supported of course, by the functions, or recipe fragments we have been building. We peeked ahead, breaking with Pascal tradition, becuase, after looking at some low-level ingredients, I thought it imporant to see where we are going at this point. You can see the value of a single-line program. It now can be moved around in your operations plan at will. You may serve your users with the frequency and taste they demand. Note, at this point, you won't have much code to change if you wanted to serve your A-M diners at 10 minute intervals beginning at 5 after the hour and your N-Z diners on the 10-minute marks.
So what does “user_notify” look like? I toiled with this one. Let me share the trials. First I did this:

```
    function user_notify { do_notify $(num_msg $1) $1; }
```

thinking that if I first calculated the number of messages for the user, and supplied that number and the user name to the function, then that function (do_notify) could perform the decision and send the message. Before going further, we have to digress. In the Korn shell, which I use excusively, the result of the operation in the expression: $( ... ) is returned to the command line. So, in our case, the result of “num_mag {user_name}” is a number 0 through some positive number, indicating the number of mail messages the user has waiting.
This version of user_notify would expect a “do_notify” to look like this:

```
function do_notify { if [ "$1" -gt "0" ]; then notify_user $2 $1; f
```

This is OK, but it means yet another “notify” function, and even for this one-line fanatic, that's a bit much. So, what to do? Observe, the only useful piece of information in this function is another function name “notify_user”. This is where culinary art, inspiration, and experience come in. Let's try a function which looks like this:

```
function foo { { if [ "$X" -gt "0" ]; then eval $*; fi }
```

This is different than the “do_notify” we currently have. First of all, $X, is not an actual shell variable, but here the X stands for “lets see what is the best argument number to use for the numeric test”. And the “eval $*” performs an evaluation of all its arguments. And here's the spice that gives this whole recipe it's flavor! The first argument may be another command or function name! A remarkable, and little used property of the shell is to pass command names as arguments.
So, let's give “foo” a name. What does it do? If one of its arguments is non-zero, then it performs a function (it's first argument) on all the other arguments. Let's solve for X. It could be any of the positional parameters, but to be completely general, it probably should be the next one, as it's the only other one this function ever has to know about. So, let's call this thing:

```
if_non_zero {function} {number} ....
```

Using another convenient shorthand, it all becomes:

```
function if_non_zero { [ $2 -gt 0 ] && eval $*; }
```

and we'll see how it's used later. With this function, user_notify now looks like:

```
function user_notify { if_non_zero do_notify $(num_msg $1) $1; }
```

and is used: user_notify {user_name}. Note the dual use of the first argument, which is the user's name. In one case, it is a further arguement to the num_msg function which return the number for that user, in the other case, it merely stands for itself, but now as the 2nd argument to “do_notify”. So, what does “do_notify” look like. We've already written the sub pieces, so, it's simply:

```
function do_notify { you_have_mail $1 | to_samba $2; }
```

At this point, we have (almost) all the recipe ingredients. The careful reader has noted the omission of “but_not”, “linecount”, and “foreach”. Permit me another gastronomic aside. Ruth Reichel, recently food editor of the New York Times, is now the editor for Gourmet magazine. One of the things she promises to do is correct the complicated recipes so frequently seen in their pages. For example, “use 1/4 cup lemon juice” will replace the paragraph of instructions on how to extract that juice from a lemon.
In that spirit, I'll let you readers write your own “but_not” and “linecount”. Let me show you the “foreach” you can use:

```
        function foreach { cmd=$1; shift; for arg in $*; do eval $cmd $a
```

A slightly more elegant version avoids the temporary file name:

```
        function foreach { for a $(shifted $*); do eval $1 $a; done; }
```

which requires “shifted”:

```
        function shifted { shift; echo $*; }
```

The former “foreach”, to be completely secure, needs a “typeset” qualifier in front of the “cmd” variable. It's another reason to avoid the use of variable names. This comes under the general rule that not every progamming feature needs to be used.
We need one final “Chapters of the Cookbook” instruction before putting this recipe back together. Let's imagine by now, that we are practicing, student chef's and we have a little repertoire of our own. So what's an easy way to re-use those cooking tricks of the past. In the programming sense, we put them in a function library and invoke the library in our scripts. In this case, let's assume we have “foreach”, “but_not”, and “linecount” in the cookbook. Put that file “cookbook” either in the current directory, but more usefully, somewhere along your PATH variable. Using Chez Marcel's example, we might expect to put it in, say, /usr/local/recipe/cookbook, so you might do this in your environment:

```
     PATH=$PATH:/usr/local/recipe
```

and then, in your shell files, or recipes, you would have a line like this:

```
      . cookbook    #  "dot - cookbook"
```

where the “dot” reads, or “sources” the contents of the cookbook file into the current shell. So, let's put it together:

```
  # -- Mail Notification, Marty McGowan, mcfly@workmail.com, 9/9/99
  #
    . cookbook
  # ----------------------------------------- General Purpose --
  function if_non_zero { [ $2 -gt 0 ] && eval $*; }
  function to_samba { /usr/bin/smbclient -M $1; }
  # ------------------------------------- Application Specific --
  function num_msg  { frm $1 | but_not "Mail System Internal Data" | lin
  function you_have_mail  {
     echo "You have $1 e-mail message(s) waiting."
     echo " "
     echo "Please start your e-mail client to collect mail."
  }
  function do_notify   { you_have_mail $1 | to_samba $2; }
  function user_notify { if_non_zero do_notify $(num_msg $1) $1; }
  #
  # ---------------------------------------- Mail Notification --
  #
    foreach user_notify  'cat /usr/etc/local/mail_notify'
```

On closing, there are a few things that suggest themselves here. “if_non_zero” probably belongs in the cookbook. It may already be in mine. And also “to_samba”. Where does that go? I keep one master cookbook, for little recipes that may be used in any type of cooking. Also, I keep specialty cookbooks for each style that needs its own repertoire. So, I may have a Samba cookbook as well. After I've done some cooking, and in a new recipe, I might find the need for some fragment I've used before. Hopefully, it's in the cookbook. If it's not there, I ask myself, “is this little bit ready for wider use?”. If so, I put it in the cookbook, or, after a while other little fragments might find their way into the specialty books. So, in the not too distant future, I might have a file, called “samba_recipe”, which starts out like:

```
# --------------- Samba Recipes, uses the Cookbook, Adds SAMBA --
. cookbook
# ------------------------------------------- General Purpose --
function to_samba { /usr/bin/smbclient -M $1; }
```

This leads to a recipe with three fewer lines and the cookbook has been replace with “samba_recipes” at the start.
Let me say just two things about style: my functions either fit on one line or not. If they do, each phrase needs to be separated by a semi-colon (;), if not, a newline is sufficient. My multi-line function closes with a curly brace on it's own line. Also, my comments are ”right-justified“, with two trailing dashes. Find your style, and stick to it.

In conclusion, note how we've eliminated temporary files and variables. Nor are there nested decisions or program flow. How was this achieved? Each of these are now “atomic” actions. The one decision in this recipe, “does Marcel have any mail now?” has been encapsulated in the “if_non_zero” function, which is supplied the result of the “num_msg” query. Also, the looping construct has been folded into the “foreach” function. This one function has simplified my recipes greatly. (I've also found it necessary to write a “foreachi” function which passes a single argument to each function executed.)

The temporary files disappeared into the pipe, which was Unix' (linux) single greatest invention. The idea that one progam might read its input from the output from another was not widely understood when Unix was invented. And the temporary names dispappeard into the shell variable arguments. The shell function, which is very well defined in the Korn shell, adds greatly to this simlification.

To debug in this style, I've found it practical to add two things to a function to tell me what's going on in the oven. For example:

```
    function do_notify   { comment do_notify $*
        you_have_mail $1 | tee do_notify.$$ |  to_samba $2
        }
```

where “comment” looks like:

```
        function comment { echo $* 1>&2; }
```

Hopefully, the chef's in the audience will find use for these approaches to their recipes. I'll admit this style is not the easiest to adapt, but soon it will yield recipes of more even consistency, both in taste and temperature. And a programming style that will expand each chef's culinary art.
—Marty McGowan, mcfly@workmail.com

Archive Index Issue Table of Contents

Advanced search

# upFRONT

**Various**

Issue #67, November 1999

*LJ* Index, Strictly On-Line and more.

## upFRONT

### THE NUMBERS

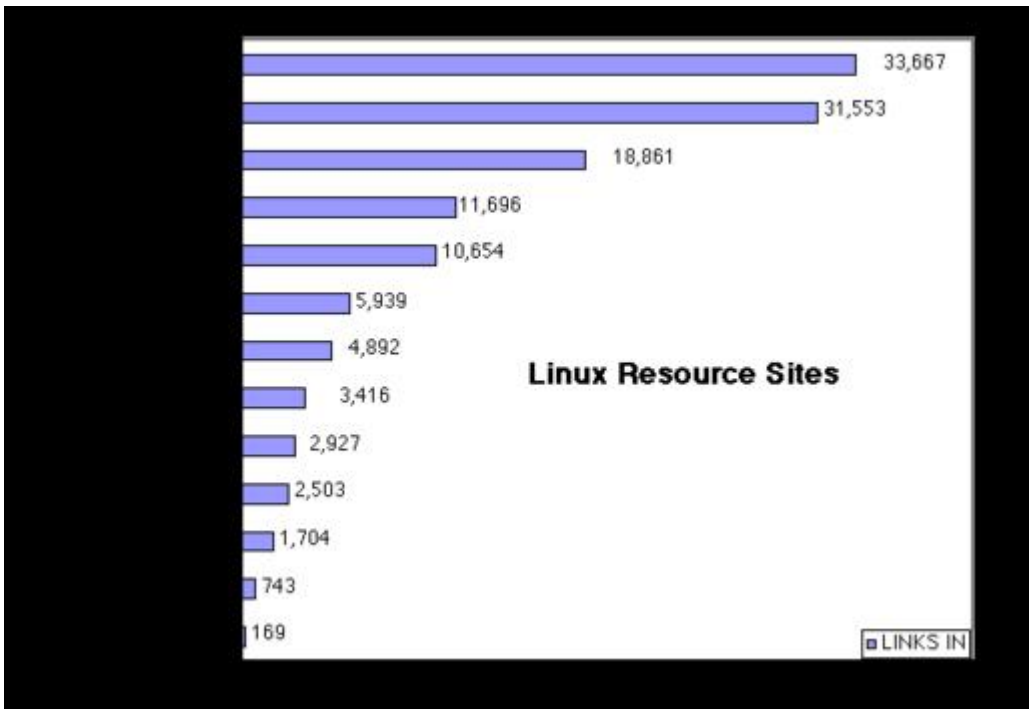
Every few months, we take another look at Alexa Internet's stats for Linux-related web sites. These charts follow the last set by 100 days, and the trends are interesting. The most noteworthy items:

- Woven Goods for Linux had the highest increase of “Links In” among the resource sites: 226%. Freecode was second, with 192%. Linux Today was third, with 154%. The increase for the whole group was 112%.
- For resource site visits, Linux.com led the way with a 164% increase. Freshmeat was second with 112%. The increase for the whole group was 18%.
- Among the Distribution sites, the leading gainer in both “Visits” and “Links In” was SuSE, with 99% and 77%, respectively. The increases for the whole category were 52% and 25%, respectively.

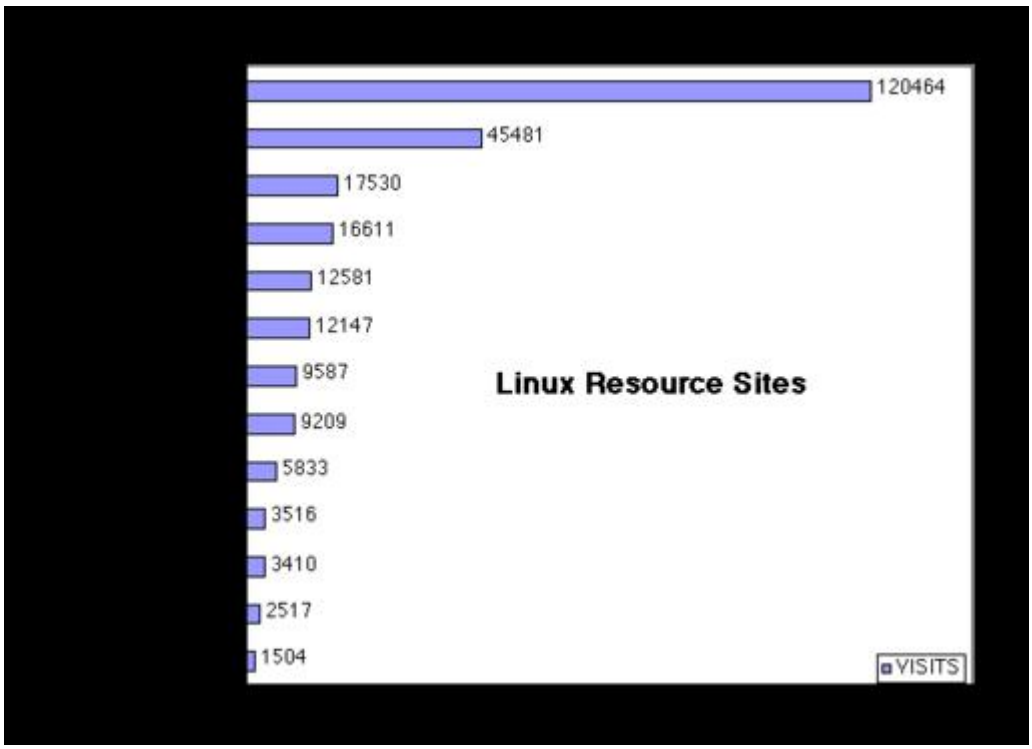
As for the rest, you can draw your own conclusions. Bear in mind the following disclaimers. These numbers were gathered on August 21, 1999. Links In are current and updated constantly by Alexa's 'bot. Visits are totaled over the preceding 6-month period. Since Alexa's client software lives only on Windows and Mac machines, visits do not (yet) include Linux or UNIX clients (see the *LJ* Index for some working irony on that subject). “SSC sites” include the *Linux Journal*, *Linux Gazette*, Linux Resources and ssc.com sites. *Linux Journal* and Linux Resources were combined during the survey period. Linux.com also started in late May and was not in existence through much of the survey period. The Slackware numbers below include all of http://www.cdrom.com/ (of

which Slackware puts up the biggest numbers). Caldera also split into two sites (and two companies) during this period.

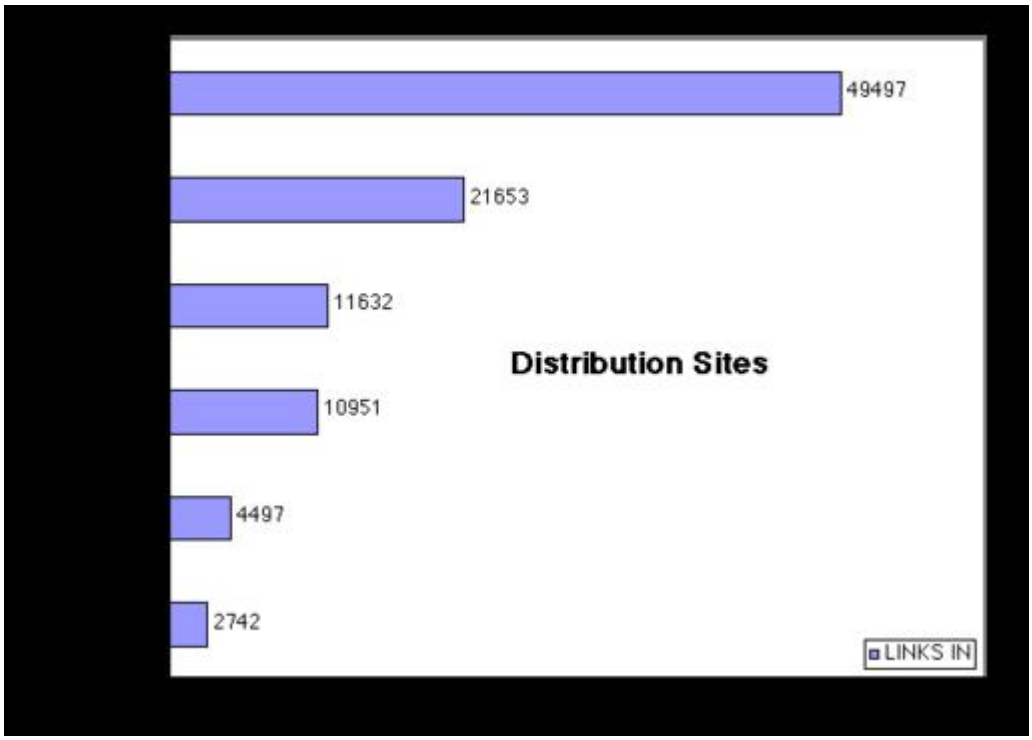
—Doc Searls



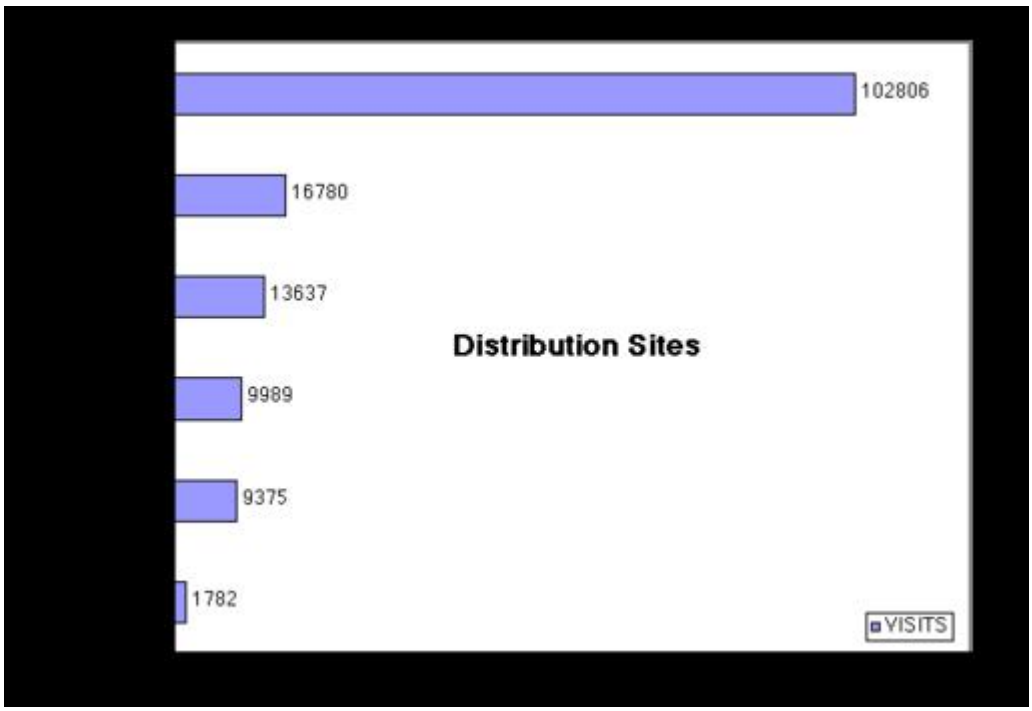

Links In for Linux Resource Sites



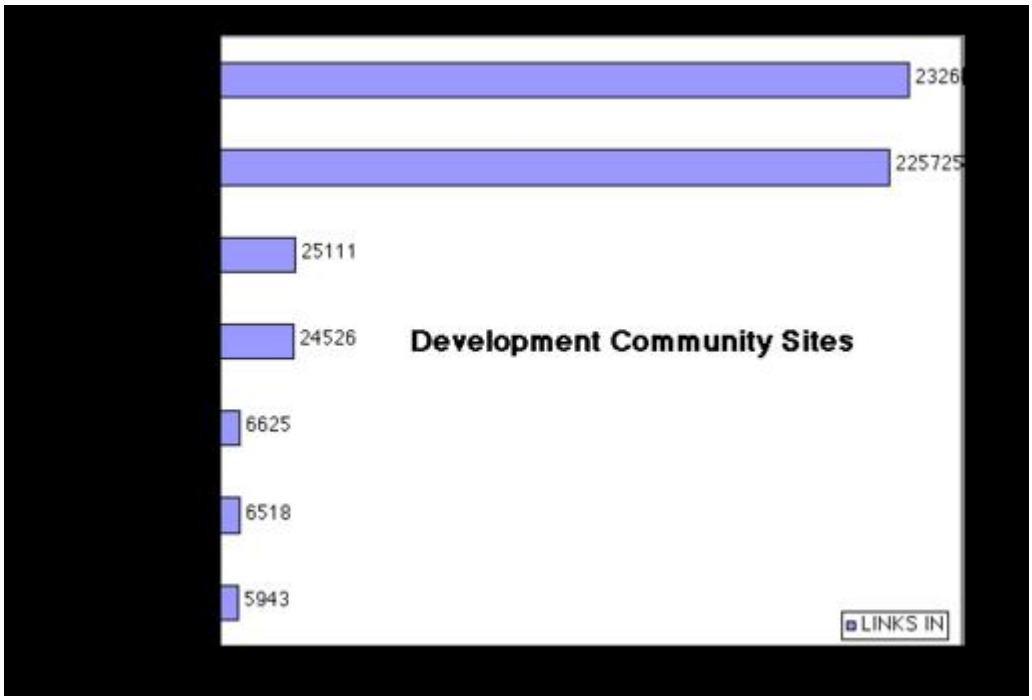

Visits for Linux Resource Sites



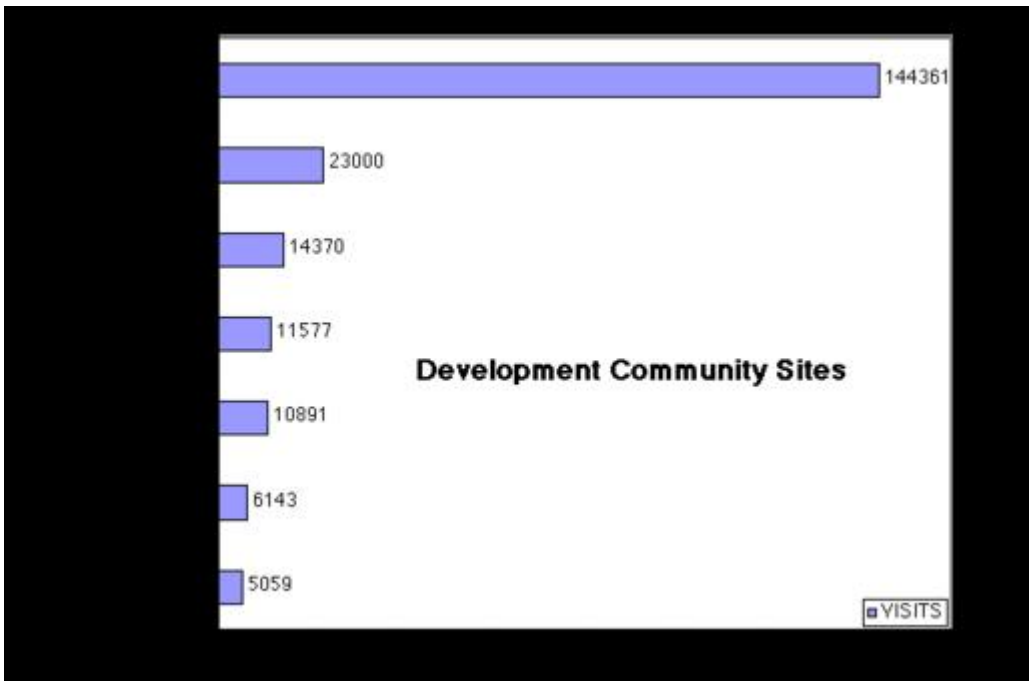

Links In for Linux Distribution Sites




Visits for Linux Distribution Sites




Links In for Development Community Sites




Visits for Development Community Sites

### *LJ* INDEX—OCTOBER, 1999

1. Number of Red Hat shares released for sale in its IPO: **6,000,000**
2. Number of shares retained by Red Hat after the offering: **66,835,104**
3. Percentage of shares in which the trading price determines 100% of Red Hat's full market value: **9%**
4. Offering price for Red Hat's IPO shares: **$14 US/share**

5. Red Hat market capitalization (market cap) at its initial offering price (all shares times price): **$935,000,000 US**
6. Top Red Hat share price within its first week on the market: **$90.69 US**
7. Red Hat market cap at its top first-week share price: **$6,061,275,581.76 US**
8. Red Hat 1999 (fiscal year ends February) sales: **$10.79 million US**
9. Red Hat 1999 net income: **-$90 thousand US**
10. Estimated size of the Linux-based server market in 2003: **$3.8 billion US**
11. Estimated Linux server licenses in 1998: **750,000**
12. Estimated non-Linux UNIX server licenses in 1998: **< 750,000**
13. Size of the Internet access market in 1999: **$2 billion US**
14. Estimated size of the Internet access market in 2002: **$16 billion US**
15. Number of web pages with links to the Linux Documentation Project: **225,725**
16. Number of Alexa Internet client visits to the Linux Documentation Project over a six-month period: **144,361**
17. Number of Alexa Internet client visits to Red Hat over the same six-month period: **102,806**
18. Percentage of Linux clients in both numbers above: **0**
19. Number of web pages in Alexa's Archive: **1.2 billion**
20. New pages added each week to Alexa: **120 million**
21. Length of time for the Web to double in size: **8-9 months**
22. Current size of the Web: **7 terrabytes**
23. Number of Red Hat shares owned by first person to achieve billionaire status as a result of Linux stock: **Frank Batten Jr.** (15 million)
24. Price per share of Red Hat stock that would make Marc Ewing and Robert Young billionaires: **$110.11**
25. Number of Penguin Peppermints eaten by *LJ* staffer David Penn, in a recent employee “contest”: **71**
26. Number of grams of caffeine per mint: 16 milligrams
27. Number of work days missed by Mr. Penn as a result: **3**
28. Number produced after converting Bill Gates III into ASCII code: **666**
29. Number produced after converting Windows 95 into ASCII code: **666**
30. Number produced after converting MS-DOS 6.31 into ASCII code: **666**
31. Amount of money donated by ex-Microsoft employee Bruce McKinney, to help legalize marijuana in Washington state: **$100,000 US**

### Sources

- Number 1-9 are from forms filed with the SEC and public stock trading data.
- Numbers 10-12 are from Dataquest, quoted in Computer Reseller News: http://www.techweb.com/se/directlink.cgi?CRN19990823S0058.
- Numbers 13 and 14 are Dataquest and Merrill Lynch numbers quoted by Motorola Computer Systems.
- Numbers 15-22 are from Alexa Internet.
- Numbers 23-24 are from *Linux Today*.
- Numbers 25-30 are from Jason Schumaker, *LJ* Staff.
- Number 31 is from *The Seattle Times*.

## HOT HAT ROAST

As we go to press, it still isn't possible to make full sense out of the Red Hat IPO. But we can report that the folks at Prosthetic Monkey Consulting (http://www.prosthetic-monkey.com/) have used some expert Python to wring maximum irony out of Red Hat's instant billions, through a web instrument called the Red Hat Wealth Monitor (http://prosthetic-monkey.com/RHWM/).

“If you use this information to flame Red Hat, then you're misusing it,” writes Kendall G. Clark, the author of the site. Bettew to lightly roast the company, which the RHWM site does rather well. To wit:

### How much is Red Hat worth?

Last updated on Wed, Sept. 15 1999 12:10:00. Updated every 15 minutes during NASDAQ trading. Trading at $96.84375 US:

| Category | Amount |
|---|---|
| Red Hat Market Capitalization | $6,472,562,103 US |
| RH Market Capitalization created by the Community | $5,650,546,715 US |
| Your contribution | $1,130,109 US |

## Who's Benefitting the Most?

| Shareholder | Number of Shares | Present Value |
|---|---|---|
| Shares Set-Aside for the Community | 800,000 | $77,475,000 US |
| Maximum Set Aside Shares Per Developer | 400 | $38,737 US |
| Bob Young | 9,081,826 | $879,518,086 US |
| Marc Ewing | 9,088,476 | $880,162,097 US |
| Frank Batten, Jr. | 15,005,888 | $1,453,226,466 US |
| William Kaiser | 8,723,866 | $844,851,897 US |
| Greylock IX Limited Partnership | 8,723,866 | $844,851,897 US |
| Benchmark Capital | 5,815,910 | $563,234,534 US |
| Intel | 3,005,058 | $291,021,085 US |
| Matthew Szulik | 2,736,248 | $264,988,517 US |
| Eric Hahn | 171,522 | $16,610,833 US |

Not bad for free software, huh?

Credit where due. “We've obviously stolen Phil Greenspun's idea for the Bill Gates Wealth Clock (http://www.webho.com/WealthClock/) and applied it to Red Hat,” Clark writes. “We share Phil's opinion about Bill Gates, but we're not anti-Red Hat. We just want to see them spreading some of their newfound wealth—which is not equal to their market capitalization—around to those who helped them acquire it; particularly in the form of R & D funds to the Community.”

We might add that on this same day (August 23rd), the Bill Gates Wealth Clock put the man's worth at $97.6121 billion US. That sum, however, was recently reduced by a chunk of change given to the Bill and Melinda Gates Foundation, making it, at $17.1 billion US, the largest charitable foundation in the United States.

So just how much did Mr. Bill give? Try $6 billion US. Or, by today's reckoning, just a few hundred million more than one Red Hat.

—Doc Searls

## TWO-QUESTION INTERVIEW

Doc Searls with Don Harbison, Marketing Manager, Notes/Domino Product Marketing, at Linux World Expo in August

**Doc Searls**: Why Linux now?

**Don Harbison**: Customer demand. When our president and CEO Jeff Papas announced at our Lotusphere event in January that we were adding Linux as the newest platform to the Domino stable of platforms, he got a standing ovation from 10,000 business partners and customers in this Disney World ballroom. It was huge. If you go to our site, you'll find that now is the first time we've made code publicly available.

**Doc Searls**: What's the difference for you, now that you're overlapping with the open-source world?

**Don Harbison**: In the open-source world, the technical community has the ability to influence the direction of the OS. So the tool kit is not hostaged by one entity. This also gives us a new model for rapid development of innovation. It's exciting!

## SIG OF THE MONTH

On Bill Gates' tombstone: “This man has performed an illegal operation and has been shut down.”--BBC Radio 4

## AND YOU THOUGHT THE NET WAS A BIG DRAW

Start-ups are all in the same business—selling promises to venture capitalists. And what a business it is. In the second quarter of this year, VCs spent $2.13 billion dollars US on 227 start-ups in the Bay Area alone. That's nearly double the money spent in the same period one year earlier. Most of the recipients were in the “Internet space”, and at least two were Linux-related. One was VA

Linux Systems, which received $25 million US. The other was Google, which received another $25 million US. (Source: *San Jose Mercury News*, Wednesday, August 11, 1999.)

If the amount of ink being spilled on a subject is a good harbinger of investments to come, look for Linux to be a prime attractor of VC money very soon. Right now, the only thing holding the VCs back is the same question that kept them from investing in those Internet companies a few years back: “If it's free and nobody owns it, how can you make money at it?”

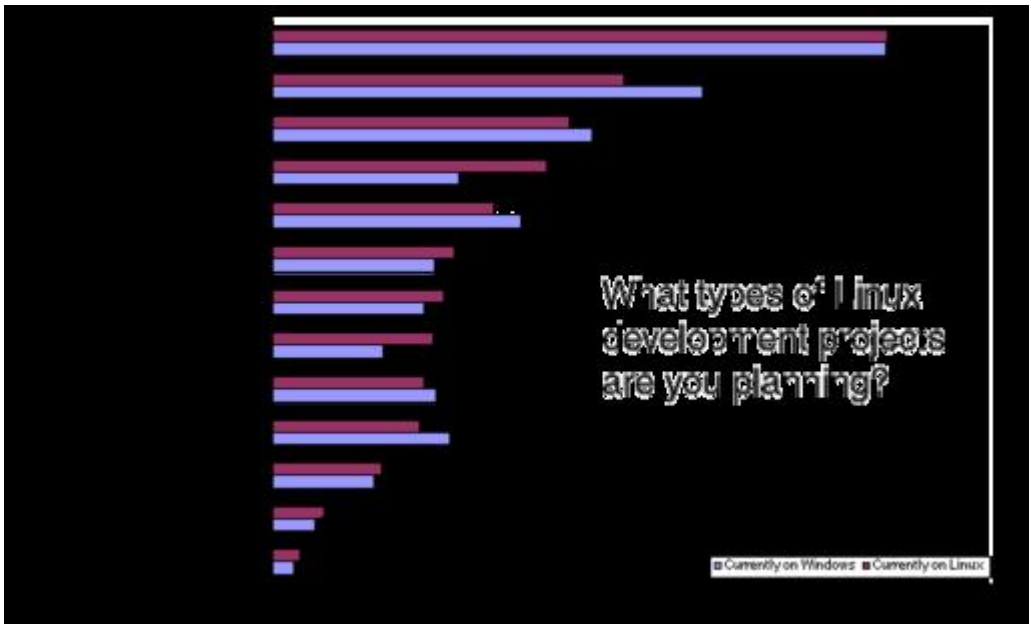
We're sure you can think of an answer.

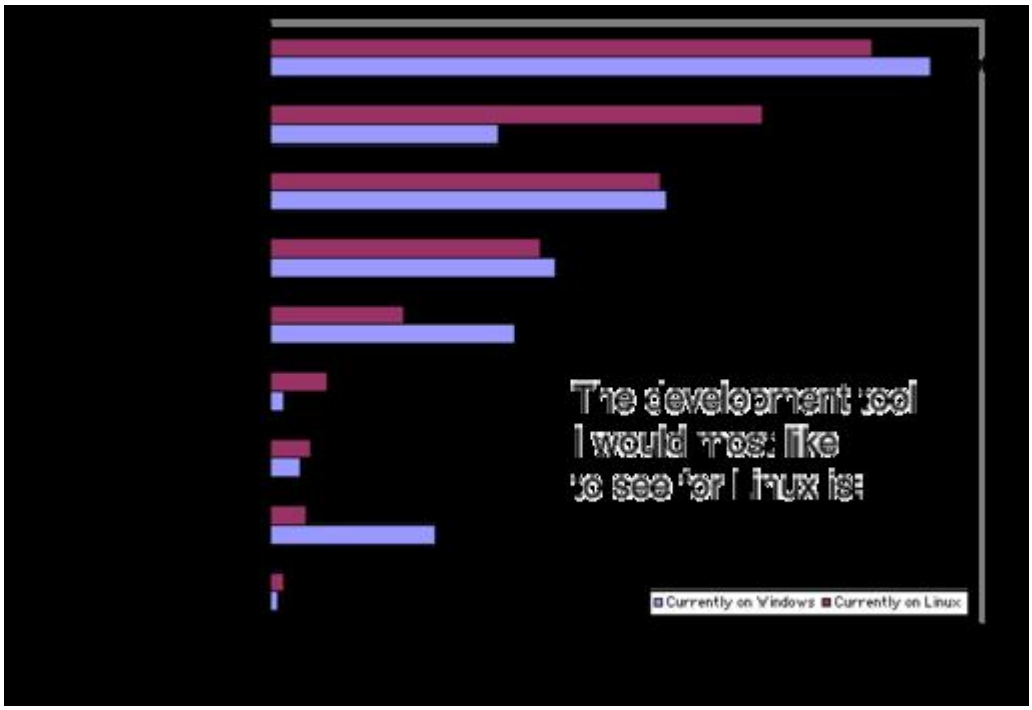
—Doc Searls

## BORLAND SURVEY: THAT'S RAD.


For most of this past July, the Borland Developer Solutions Group at Inprise ran a Web-based survey on Linux development, which they promoted from links at *Linux Journal*, *Slashdot*, *Linux Today*, SuSE and Borland.com. It was the largest survey of its type in the company's history, generating over 24,000 unique survey submissions. There are pages and pages of results, which you can read at http://www.borland.com/linux/. But none make side-by-side comparisons of responses from programmers currently using Linux or Windows as their primary development platform. The results are pretty close (hey, they both want to work on Linux). One take-away is a shared appetite for Rapid Application Development (RAD), Integrated Development Environments (IDEs) such as Borland's own Delphi, and—especially important to the Linux natives—IDEs that work with existing standard Linux tools.

—Doc Searls


What types of Linux
development projects
are you planning?
Currently on Windows
Currently on Linux


The development tool
I would most like
to see for Linux is:
Currently on Windows
Currently on Linux

EVENTS

- WebNet99, World Conference on the WWW and Internet, http://www.aace.org/conf/webnet/advprog.htm, October 24-30, 1999, Honolulu, Hawaii
- USENIX LISA, the Systems Administration Conference, http://www.usenix.org/events/lisa99, November 7-12, 1999, Seattle, WA
- COMDEX Fall/Linux Business Expo, http://www.comdex.com/comdex/owa/event_home?v_event_id=289, http://www.zdevents.com/linuxbizexpo/, Nov. 15-19, 1999, Las Vegas, NV
- SANS 1999 Workshop on Securing Linux, http://www.sans.org/, Dec. 15-16, 1999, San Francisco, CA. The SANS Institute is a cooperative education and research organization.

## EARTH-SHAKING HARBINGERS

“Well, [the IPO] certainly looks like a success! [Red Hat] raised a good bit of capital, which is going to help us scale our business, so from that standpoint, sure.”

“We offered approximately 5,000 open-source developers with an opportunity to be involved. Of that, 1,300 people expressed an interest in being involved, and of those, basically 1,150 of them were able to become Red Hat shareholders.”

—Donnie Barnes, Director of Technical Projects for Red Hat, to Marjorie Richardson, September 7, discussing Red Hat's IPO. (Complete interview can be found at http://www.linuxjournal.com/articles/conversations/004.html.)

## AND THE WINNER IS…

The man on our September cover is Terry Burkhardt, co-owner of Burk's Creole Cafe in the Ballard area of Seattle, Washington. Right around the corner from SSC's building, Burk's Cafe is a favorite place for *LJ* employees to dine. We thank Burk, a great cook and a good model, for allowing us to photograph him.

We had several entries with the correct answer and chose one at random. And the winner of the Palm Pilot is Grant Ranlett of Seattle, Washington. No surprise—he lives in Ballard.

## EASY LINUX

It is entirely possible that someone, somewhere might have wondered how soon the “Linux: For IQs above 2000” T-shirts would show up. While I still think

this has a nice ring to it, Linux has become easier and easier, while the user base has gotten larger and larger. One could know literally nothing about Linux, buy a machine from VA, Penguin, ASl, Linux Laptops, NexTrend, Pogo, Telenet, DCG, HiTech or whomever, and have it up and running in a matter of minutes. However, if someone accidentally clicked the little “terminal” icon (or, by some slip of the fingers, hit **ctrl-alt-f1**), how would he know what to do?

Cyber Station's EX Linux has the answer. This company is producing reference cards, mousepads and even software exclusively for educating Linux users. The mousepad, for example, houses dozens of key Linux commands for so many important things that practically everything is covered. EZ Linux's software is an electronic reference card, quite expanded from the mousepad, and full of all kinds of things that will help you do anything you want on Linux.

If you're past the point of needing a reference card to help you mount and repair file systems, configure X and set up your network, but you can't yet coordinate yourself well enough to buy a Linux CD or do an ftp install, fezbox.com has the answer.

fezbox.com is an automated Red Hat installer on the Web. It's just like installing from a CD-ROM, only the interface sits on the Net and you get your files from an NFS server. It's still beta, and you have to specify an NFS server, but there's probably a future in web-based software installation. Maybe other distributions would like to be represented on-line in this way.

Linux has been getting easier ever since its inception, and now that the user base has expanded, it seems there is a demand to make things even less difficult. Not so long ago, we saw the birth of RPM, KDE, GNOME, easy auto-probing installers and numerous XF86 configuration utilities. Today, even kernel recompilation has an easy-to-use graphical interface. Really, what's next?

—Jason Kroll

## GAMES FOCUS-->CODE WARS

Artificial intelligence is one of my favorite topics in computers—at the very least, it's a science-fiction fantasy which inspires the imagination more than kernel development. A particularly exciting subset of AI is robotics, which, when coupled with some creative mishaps, leads to futuristic happenings as seen in countless stories, films, computer games and toys. Computer people often know the movies (maybe line for line), and probably remember the computer games (such as Andrew Braybrook's immortal Paradroid). Maybe we even had toy robots as children, though disappointingly, they didn't go crazy and take over the house/neighborhood/school/planet nearly as often as we might have liked.

Hacking, <\#224> la Hollywood, is another topic we've seen treated in books, movies and computer games. The mystique of dark hackers, clad in black, hiding in basements while performing nefarious deeds and drinking Jolt (although I have it on good authority that real hackers drink fruit juice) probably draws many people in (but what to do once we've dressed ourselves in black and gone to the basement with soda pop is quite another matter). If we can't start a global thermonuclear war (just *try* playing “wargames”), at least we can write a virus. Failing that, we'll have to play computer games.

Fortunately for the Linux-using, aspiring ne'er-do-well, there are a number of games available which allow us to do dark and mysterious things like program robots to blow each other up and code viruses to battle for RAM domination.

### Core Wars

Core Wars Screenshot

Virus enthusiasts might enjoy this game, first developed by the well-known computer scientist and author A.K. Dewdney in the early 1980s as discussed in *Scientific American*. In Core War (or Core Wars, as you like), two or more assembly code programs battle against each other in memory with the purpose of killing off all the enemies' processes by forcing them to execute a certain command. The assembly code is called Redcode, which is rather simple yet wholly adequate, and the virtual computer which runs the cores is called MARS (Memory Array Redcode Simulator). Memory arrays can range from tiny to enormous, and several cores can battle at a given time.

Like many pursuits, learning to play is simple, while strategies quickly become clever and complex. I spent a great part of my thirteenth year of life on this game, carrying a calculator with me to work out algorithms, so be warned—it can become an obsession. Also, it's good for young people, an easy way to teach assembly code to kids (well, you can't let them grow up without knowing asm, can you?).

The standards of Redcode have changed significantly since the 1980s (the initial version ultimately became rather exhausted) and now allow for much more complex programs, although the basic strategies remain the same. There are on-line tournaments running all the time, so you can try your cores against the world's best. A while ago, a friend found some of our old cores on a *disk* (which should tell you how old they are) and I tried them against the net-warriors—our cores got terminated, so it appears the standards have improved. Assembly-code devotees should do very well at this game.

### RealTimeBattle

RealTimeBattle Screenshot

If robots inspire your fancy more than viruses, RealTimeBattle by Erik Ouchterlony and Ragnar Ouchterlony may be more to your liking. Essentially, you program a robot to battle other robots in real time. One thing that distinguishes it from many programming games is that you can use any language you like to program your robots, since they communicate to the server through STDIN and STDOUT. The use of real languages allows for more legitimate intelligence in the machinery.

Robots are physically identical and have energy, scanners, cannons and movement devices. The world in which they operate has gravity, friction, air resistance, bouncing, skidding and the like. Combat arenas can be large and open or filled with walls, and can accommodate up to 120 simultaneous robots.

Though already entirely playable, this software is in a state of continual development. Already I can see many possibilities for this one, such as an on-line server, massive multi-level maze-like arenas and so on. Anyone who remembers games like Crobots on the Amiga or RobotBattle on the unmentionable OS will probably find this game quite an improvement.

### Automata Zero

There comes a time in everyone's life when a squadron of robots isn't enough. You want an army of robots, designed and programmed by you. Fine, with Automata Zero, you can have them—literally an army. And, you not only design and program the robots, you can program them to respond to real-time commands from your terminal, and you can change code on the fly. Automata Zero's graphics are much more advanced than those of Core Wars or RealTimeBattle and feature aesthetic landscapes with grass, water, trees and futuristic mechanical structures. The robots themselves look like Mechs. The development team says graphics are a low priority, but that's hard to believe when they look so nice. Zed is the name of the programming language, which doesn't feature dynamic memory allocation, floating-point operations, threads or pointers, but does have arrays, functions and scoped variables; it is a fairly simple mixture of C, BASIC and Matlab. Automata Zero is still under development and needs more developers. You can visit Automata Zero's home page at http://www.andrew.cmu.edu/~sand/ to learn how to contribute or just play the game. It's a little closer to your typical strategic conquest-of-the-world game than the others. Presumably, if you lose, the world will be overrun with enemy robots who will come to your basement and blow you up, along with whatever soda pop you may be harboring. Good luck! (May the bitstream be with you?)

—Jason Kroll


Stop the Presses: So why is Red Hat worth more than Laos?

Six months ago, Red Hat was a “free” software company that lost money on less than $11 million in sales in its last fiscal year. By late summer, it was worth upwards of $7 billion US. (Perspective: the gross domestic product of Laos is less than $6 billion US, according to the World Factbook.)

Don't waste your time wondering if Red Hat's valuation is “real” or deserved. We're talking about a real market here: the market for shares of a company. If that market says the smallest piece of Red Hat is worth exactly $100, well, it is. Multiply that by all 66,835,105 equal-sized pieces of Red Hat, and you've got a company that's worth exactly $6,683,510,400 US. That's the obvious part.

What's not so obvious is that only 6,000,000 of those shares were put on the market in Red Hat's IPO (initial public offering) in August. That's less than 10% of all the stock issues by the company. The rest were not released to the market. When there's a lot of demand and highly constrained supply, the price goes up. And with that price goes the value of *all* the company's shares. Neat, huh?

Now think about all the other Linux-related IPOs that should be coming up. We learned about Cobalt Networks and Andover.net in September, when there were also rumors about Caldera and VA Linux Systems. If these companies do even half as well as Red Hat, think of what that will do for the value of every Linux business.

Welcome to the New World!

—Doc Searls

## STRICTLY ON-LINE

**NoSQL Tutorial** by Giuseppe Paterno presents a good, long introduction to this open-source database for small- to medium-sized tables. NoSQL is a database system that was designed to get the most from a UNIX system, using commands that combine various standard tools.

**Interfacing Relational Databases to the Web** by Will Benton offers the instructions for building a database-backed web site using Apache, the PHP3 module and the PostgreSQL relational database. Information on installing the tools, setting up PostgreSQL, PHP3 basics, database connectivity and design considerations is included. A simple guest book program is used as an example.

**The ACEDB Database System** by Joe Nasal describes how to make easy work of local area network management using Linux and the object-oriented database, ACEDB. The article covers not only the ACEDB basics, but also gives information on Arpwatch, a program that collects network data, and AcePerl, the Perl interface to ACEDB.

**How to Install and Configure Oracle** by Greg Flannery gives a step-by-step walkthrough of the Oracle installation program, including all the nitty-gritty details. If you want to use Oracle on Linux, this article will show you the way.

**Linux in Education: Buy One, Get One Free** by R. Scott Gray, Luke Pargiter and Phil Pfeiffer tells us how the Computer Science Department at East Tennessee State University configured Linux as a dual-purpose user environment. The two needs were general classroom use and experimental laboratory use. This article explains how they set up the network to allow the students to experiment with the system source code without interfering with others.

**Mastering Linux** is reviewed by Bob van der Poel. This very large book was written by Arman Danesh and published by SYBEX Inc. Read the review to find out what it covers and if it is a book you need.

**Linux in the Tropics** by Rodrigo Bernardo Pimentel gives us the scoop on Linux user groups in Brazil—what they are accomplishing and how they are growing in popularity. See the web page for the National Group they have set up at http://www.linuxsp.org.br/.

**Linux Conviguration & Installation, Edition 4** by Bob van der Poel is a book review of the latest edition of this classic book. Do you need a new copy? Read the review to find out.

## VENDOR NEWS

**Amdahl Corporation** (http://www.amdahl.com/) announced it will provide full enterprise-class support for the Linux operating system on the Intel architecture-based Fujitsu teamservers. This support includes factory installation of the Linux operating system on Fujitsu single and dual Pentium III processor model teamservers, and professional services for setup and customization of Linux and associated open-source products such as the Apache web server.

**Rogue Wave Software** (http://www.roguewave.com/), a provider of software for creating enterprise systems with object-oriented component technology, announced support of the Linux platform. The company will start by shipping versions of NobleNet RPC, Tools.h++, DBTools.h++, Standard C++ Library and Money.h++.

**Debian** (http://www.debian.org/) has, as the result of a GIMP Logo contest (http://contest.gimp.org/), chosen a new logo. The logo was selected by a vote of the Debian developers. The winning entry was submitted by Raul M. Silva of onShore, Inc. and can be seen on the main Debian web pages. Licenses and conditions for use can be found at http://www.debian.org/logos/.

**Hummingbird Communications Ltd.** (http://www.hummingbird.com/), an enterprise software company, announced it has entered into a co-marketing and strategic development alliance with **Caldera Systems, Inc.** (http://www.caldera.com/), a provider of Linux-based business solutions. The new relationship will give Caldera's Linux users access to all of Hummingbird's desktop client connectivity products including Exceed, HostExplorer, NFS Maestro Server, NFS Maestro Client, NFS Maestro Gateway and NFS Maestro Solo.

**Terra Soft Solutions** (http://www.blacklablinux.com/) announced the formation of a reseller relationship with **MacMall** (http://www.macmall.com/) for the purpose of selling Yellow Dog Linux software (http://www.yellowdog.com/). The MacMall catalog has a circulation of up to two million, reaching end users, small to medium businesses, government and education markets.

**Deja.com** is the new name for DejaNews. This site is a consumer information exchange, providing consumers with the opportunity to consult the opinions and experiences of other consumers regarding specific products and services. Deja.com rebuilt their site using TowerJ from **Tower Technology** (http://www.towerj.com/). According to independent industry benchmarks, TowerJ on Linux/Intel is the performance-leading Java deployment platform (see http://www.volano.com/report.html).

**Network Concierge** (http://www.nc4u.com/) and **Motorola Computer Group** (http://www.mcg.mot.com/) announced a partnership to deliver networking solutions. Powered by Network Concierge software, the SLX series offers Motorola OEM customers reliability and service. The SLX network appliances require no operating system, networking or server expertise and can be deployed in less than 15 minutes.

**Linux System Solution Ltd** (LSSL) and **INFOMATEC-IGEL Asia Ltd** have formally announced their cooperation in the Internet device, Thin Client and Networking arena. IGEL Asia LTD (http://www.igelasia.com/) will endeavor to market and educate customers toward a complete Linux-based Thin Client/Server solution based on TurboLinux's products complementing IGEL's Firmware. LSSL will help support and develop application solutions and services in the use of Thin Clients for corporate, commercial and educational use. LSSL and IGEL will cooperate in the localization and development of Chinese applications and solutions.

**Linuxcare, Inc.** (http://www.linuxcare.com/), a provider of technical support, consulting, education and product certification for all distributions of Linux, announced it has reached an agreement with IBM to offer support services and training for the company's RS/6000 servers. In addition, Linuxcare Labs, the product certification arm of Linuxcare, entered into an agreement with Macmillan Computer Publishing to provide technical editing and certification services. **Linuxcare** also announced it had reached agreement with Sun Microsystems, Inc. (http://www.sun.com/) to provide a range of enterprise-class services for Sun's recently acquired StarOffice productivity suite on Linux. Under the agreement, Linuxcare will provide a comprehensive suite of enterprise services for StarOffice on Linux. These will include end-user and developer technical support, custom development, enterprise integration, migration and rollout, and customized training.

**sourceXchange**, the open-source development marketplace, went live on August 10 (http://www.sourceXchange.com/). At that time, more than 1500 developers had already registered with sourceXchange. Seven RFPs were presented from founding sponsor Hewlett-Packard. sourceXchange provides a dynamic forum where sponsors who need open-source programmers can contract with qualified developers for specific development projects. Developers can also submit proposals to the sourceXchange Wish List, soliciting the talents of their peers in the Open Source community for projects that contribute to the common good.

**The White Camel Awards** were presented to three leading activists in the Perl community at O'Reilly's Perl Conference 3.0 in Monterey, CA on August 23. Tom Christiansen, Kevin Lenzo and Adam Turoff were recognized for their

extraordinary contributions to Perl Advocacy, the Perl Community and Perl User Groups, respectively. The White Camel Awards were created to honor individuals who devote remarkable creativity, energy and time to the non-technical work that supports Perl's user community. The awards are sponsored by Perl Mongers, http://www.pm.org/, a not-for-profit organization whose mission, along with O'Reilly and sourceXchange, is to establish Perl user groups.

Factoid: If you care to send Linus a gift, why not send him some Guiness, his favorite beer, which he only drinks from a can! Send to *Linux Journal*, c/o Linus Torvalds :)

Archive Index Issue Table of Contents

Advanced search

# Penguin's Progress: Hacking an Industry

**Doc Searls**

Issue #67, November 1999

And they asked me how I did it, and I gave 'em the Scripture text, `You keep your light so shining a little in front o' the next!' They copied all they could follow, but they couldn't copy my mind, And I left 'em sweating and stealing a year and a half behind. ==Rudyard Kipling

In September, I suggested that the real Linux “threat”—the true course of World Domination—was not an epic battle between Linux and Windows, but the quiet transformation of the whole software industry from one known mostly for its vendors to one known mostly for its builders.

Since then, usage for the verb *build* and the noun *builder* has shot up. True, web work has always had its construction and real estate metaphors: we *build* or *develop* a *site* with an *address* or a *location*. But, to borrow another real estate metaphor, the ground is beginning to swell. In fact, it feels like there might be a volcano under there.

One harbinger is Borland (known less well these days as Inprise, or Borland/Inprise, or something like that). The company already had Builder-branded tools (J Builder, C++ Builder) when it detected the obvious fact that the server side of the Web was being built by a huge and growing number of Linux hackers. So the Borland folks thought they'd do a little research. “Would Linux developers like some Borland tools for Linux?”, they asked on a half-dozen popular Linux-related web sites. The answer was a word balloon big enough to fill the sky. Twenty-four thousand programmers spoke with one voice, “Yes!” Significantly, most of those speaking called Windows their current development platform. (Read more about it in this month's “upFRONT” section.)

Even CNET's Builder.com site and Builder.com.live! trade show, for all their Windows defaults (they still ghetto Linux discussions into a “Project Heresy” section) can't ignore developments that are manifest in their own discussion

groups: there are more than a thousand posts, talking about the two most recent Red Hat versions.

The real magma in this volcano is demand. Inprise/Borland CEO Dale Fuller says:

> The whole business world is moving to the Net. The new skyline of business is being built on the Web. And constructing it will take an enormous amount of work. Lots of builders will need lots of tools and construction materials, not to mention architectural blueprints. That's our business, and we think it's a good one to be in.

Look at the numbers. According to International Data Corp., businesses bought $80.4 billion US worth of goods and services over the Web in 1999, and consumers spent another $31 billion US. IDC expects those numbers to reach $1.1 trillion US and $177.7 billion US by 2003—increases of 1,418% and 574%, respectively. That's in cash money—for real stuff. Remember how the big revenue model for the Web was going to be advertising? Well, that will explode too, but to “just” $33 billion US in 2003, IDC says.

Now think about the infrastructure involved here. *Huge* doesn't cover it. Those numbers are just for e-commerce. What's it going to take to build out the infrastructure behind all that? We know it'll take two things for sure: Linux and Apache—two well-proven building materials. Of course, Windows 2000 will also be involved. There are just too many people already constructing this new skyline with Microsoft tools and building materials. The difference is that the builders themselves help improve Linux, Apache and other open-source products. They can't do the same for Microsoft.

One developer put it to me this way:

> When I'm building a skyscraper, I want to know there's rebar in the concrete. With Linux, I know. With Microsoft, I don't. In fact, NT's memory leaks prove to me there isn't rebar in there. Since I have to work with NT for political reasons, I just cope with it. But I know if we could see the source, we could probably fix the problem pretty fast.

It's a subtle thing, but I feel the center of gravity in the software industry starting to shift from platforms and applications to tools and building materials. Or, as Eric Raymond likes to put it, from sale value to use value. The irony is, use value helps make a bigger overall industry.

So the next question is: If the software industry is going to turn into another construction industry, what becomes of Microsoft? To help find that answer, let's ask: who is the Microsoft of the construction industry?

Is it Home Depot? With more than $8 billion US in 1999 sales, Home Depot is the “big box” store for the do-it-yourself business, and a “category killer” for hardware stores. But technically, it's a retail business.

The biggest home builder in the U.S. is Centex, with more than $5 billion US in 1998 sales. Behind Centex are Pulte and Kauffman & Broad, both in the $2.5 billion US range. But none of those companies are household names. Equally unfamiliar is Japan's Shimizu, which outweighs all three American leaders. Even less memorable (and pronounceable) is ABB Asea Brown Boveri of Switzerland, which had nearly $31 billion US in 1998 sales. That's not only bigger than all those other construction companies, but far ahead of Microsoft, which had less than $20 billion US in the same year.

The big difference, of course, is that Microsoft's earnings—its profits—approached $8 billion US, while ABB Asea Brown Boveri barely passed $1.3 billion US. At almost 97%, Microsoft's gross profit margin was well ahead of the industry average of 82%, and more than double the average for the eight thousand members of the New York Stock Exchange. Its net profit margin (39.4%) also more than doubled its own industry (15.8%) and was more than seven times larger than the average for all of Wall Street (5.8%).

As I write this, Microsoft has a market value of around half a trillion dollars. Compare that to General Motors, which is number one on the Fortune 500 in 1998 sales of more than $161 billion US, but which earned less than $3 billion US in the same year and currently has a market value of less than $50 billion US. Microsoft is simply the largest, most profitable and most durable member of the world's most profitable major industry.

Let's put this in perspective. The total worldwide packaged software market—Microsoft's category—was $135 billion US in 1998, and growing at a 13.6% rate. But the worldwide outsourcing services market was nearly as big: $100 billion US in 1998 and growing at about the same rate. The computer services industry was $90 billion US in 1997. And we're not even touching a raft of other hefty categories: enterprise resource planning, security, database management, transaction systems and so on.

Meanwhile, the construction industry in the U.S. alone is $619 billion US. Yet the leading builder, Centex, accounts for less than 1% of the whole industry. That's because most of the industry is local. “Compared to the globalization driving much of the construction industry, the localized nature of home

building seems almost quaint. Small local contractors still account for four-fifths of all domestic residential construction", writes Hoovers (http://www.hoovers.com/). And that's just counting the current job-holders. The do-it-yourself business is also huge, as the success of Home Depot shows.

And that brings us back to what's really going on.

Linux and other open-source products—along with constantly improving commercial tools from the likes of Borland/Inprise—are equipping software architects, developers and solution builders to take over the whole software business. If it follows the lead of the construction business, the result will be a much bigger pie for everybody to split, including Microsoft.

I think we'll see the software industry start to explode on the same curve, and on the same scale as the last "free" development everybody but the hackers ignored—the Internet.

Archive Index Issue Table of Contents

Advanced search

# Best of Technical Support

**Various**

Issue #67, November 1999

Our experts answer your technical questions.

## Synchronizing Clocks

I would like to know how best to synchronize the clocks of the various Linux boxes (various kernels and distributions) on my LAN. —John Morley, jmorley@grafton.demon.co.uk

You should try NTP: www.eecis.udel.edu/~ntp. NTP software is included in common Linux distributions such as Red Hat. —Pierre Ficheux, pficheux@com1.fr

I personally run **rdate** from my computer's crontab. This gives me an accuracy within one second, which is satisfactory for normal interactive use of the net.—Alessandro Rubini, rubini@prosa.it

This is definitely a useful thing to do wherever you have more than one server, but there are many ways to accomplish it. The most common is to use the **timed** daemon. Read the man page of the daemon for a detailed description of how to use it. FYI, there are other programs you can download that will do the same job with slightly different features, but since timed is present on almost all modern UNIX systems, it's the best place to start. —Chad Robinson, chadr@brt.com

## Kernel, Modules, Mouse

I am using SuSE Linux 6.1. I defined a new kernel configuration (using **make xconfig**). If I enable “modules” support, my mouse does not operate. Using exactly the same configuration but creating “monolithic” kernel (meaning that I changed all m's into y's) works fine. I spent a lot of time on this problem and got no results. Do you have any ideas? —Mark Shahaf, mshahaf@ibm.net

You likely have a problem with module autoloading. Try loading the module for your mouse by hand: **modprobe psaux** for example. Also, if the problem affects only your mouse, you can use modules but compile it in the kernel. Since you probably use the mouse all the time and the code isn't too big, you might as well compile it in the kernel. —Marc Merlin, merlin@varesearch.com

If the mouse driver is compiled as a module, you should load the module in order to be able to use the mouse. Try **insmod psaux** (or **insmod serial** if it is a serial mouse), or configure **kmod** to load it automatically. —Alessandro Rubini, rubini@prosa.it

### startx and Colormaps

I am running Red Hat Linux 6.0. If I run in full multiuser mode (by setting the run level in the inittab file to 3), I can execute **startx - -bpp 32** and have all the necessary colors displayed. However, if I change the run level from 3 to 5 in the inittab file to run X11 automatically, I receive the following results when I try to load some GUI applications:

```
Warning: Cannot allocate colormap entry for "gray71"
Warning: Cannot allocate colormap entry for "AntiqueWhite3"
Warning: Cannot allocate colormap entry for "AntiqueWhite2"
...
```

This happens even though I have already modified the last line in the /etc/X11/xdm/Xservers as follows:

```
:0 local /usr/X11R6/bin/X -bpp 32
```

Any suggestions please? —Mohsen Madi, mmadi@cs.umanitoba.ca

This is exactly what you are supposed to do if you use **xdm**. However, if you use **gdm** (the default in Red Hat 6.0), you now need to edit /etc/X11/gdm/gdm.conf. You will find this section:

```
[servers]
0=/usr/bin/X11/X
```

Just add your **-bpp 32** there. —Marc Merlin, merlin@varesearch.com

### Another startx Problem

When I type **startx** as a regular user, after the initialization of card, I receive this response:

```
X11TransSocketUNIXConnect : Can't connect: errno = 111
giving up
xinit: Connection refused(errno 111) unable to connect to X server
xinit: no such process(errno 3) server error
X11TransSocketUNIXConnect : Can't connect: errno = 111
```

I am running XF86 3.3.3.1-49. I found this very same post in the magazine I get; however, mine is a bit odd. I can run X as root with no problem. I have permissions set for all users to execute X, but things aren't going that way.

On another note, I downloaded the bz2 package of 2.2.10 and installed it. I added support for the /proc file system, for my Linux box to act mainly as a router, for forwarding, firewalling, masquerading and the extra networking goodies. However, when I rebooted, I had no /sys subdirectory under /proc. I do have the /proc directory tree, though. I can start the firewall with no problem and pass arguments to ipchains. When I run **uname -a**, I get this output:

```
Linux digitalklown.net 2.2.10 #1 Mon Jul 26 21:17:02 CDT 1999 i586 unknown
running rh6.0
```

—Jason Helfman, deklown@digitalklown.net

The reason you don't have /proc/sys is that you forgot to add **sysctl** support in the General setup section of the kernel configuration. When you face unknown problems like this, your best bet is to make sure your 2.2.10 tree is clean by typing **make distclean** (this will erase your .config file which contains all the compilation options you chose, so if your current copy is there, save it first). If you do have a .config file, now you can copy it into your 2.2.10 tree and build your kernel with the usual **make clean** (although useless here, since we just did **make distclean**); **make dep**; **make install**; **make modules**; **make modules_install**. If you do not have a .config you can get the .config Red Hat used for their kernel by installing the kernel sources by going to /usr/src/linux-version and typing **make oldconfig**. This will generate a .config in the same directory, and you'll be able to move it to your new 2.2.10 tree.

Following the above procedure should insure that you have a sane set of options selected in your kernel and that the kernel you end up with is built correctly. —Marc Merlin, merlin@varesearch.com

With **startx**, you fire both the X server and a set of default clients. The errors you report are the clients' errors, which show that the server is not running. You should look earlier on your screen to find why the server failed.

If the server is actually running, then you may have authorization problems. Please check whether your xauth setup is wrong or other authorization means are running and misconfigured. The /proc/sys tree is part of the **sysctl** implementation. If you didn't enable sysctl when configuring the kernel, no such tree is there. —Alessandro Rubini, rubini@prosa.it

### SIS Drivers Problem

I installed Red Hat Linux 5.0 on my system which has a Cirrus Logic VGA card. I installed the X Window System and am running it with success. In my office, most of the systems have SIS 6215 VGA cards, and Linux 5.0 does not support this. I downloaded SIS drivers from Red Hat's site. I am using the PC-quest Red Hat Linux 5.0 CD which does not have SIS drivers. How do I insert the SIS drivers during installation, since I am installing from CD? Please clarify my doubts on how third-party drivers should be installed without CD during installation. —Munnangi Reddy, rajasekhara_m@hotmail.com

You can't. However, the installation doesn't use graphics, so you can install the system anyway. After installing, you can upgrade your X packages by installing the new RPM file using

```
rpm -i package
```

—Alessandro Rubini, rubini@prosa.it

If you want to use new XFree86-supported cards, you should upgrade both the XFree86 server (XFree86-SVGA...) and the Xconfigurator utility, which is used in order to generate an XF86Config file. All these packages are available from the Red Hat FTP server. —Pierre Ficheux, pficheux@com1.fr

### PPP Advanced Question

I am using Red Hat version 6.0. I have PPP configured and working for dial-out to my ISP. It uses a dynamic IP address assigned by the server. I also wish to allow dial-in on the same system to allow for administration and tech support. The documentation states that you put the IP address you wish to assign to the port for dial-in users in the /etc/ppp/options.ttySx file. However, as long as I put an IP number in the file which corresponds with the dial-out port, my PPP dial-out fails. I know it is possible to support both dial-in and dynamic dial-out on the same port. The PPP HOWTOs state that it can be done. What do I have to do to get it to work?

Note: the options.ttySx file has only the single entry. All other options are in the options file. —Gerry George, ggeorge@digisolv.com

Actually, you would do this only if you have multiple serial ports and modems and you want to assign IP addresses dynamically to your users. If you have only one modem, you can simply assign the IP on the PPP command line. You could create a PPP user in /etc/password which launches this script in lieu of a shell:

```
#!/bin/sh
IFS=" "
export IFS
```

```
/usr/bin/mesg n
stty -tostop -echo
exec /usr/sbin/pppd modem crtscts proxyarp -d\
  -detach moremagic:ppp-guest
```

—Marc Merlin, merlin@varesearch.com

### Multiple Authorized Users

We have several labs of Linux boxes available for student use at the University of Arizona Physics Dept., as well as a couple of older SPARCs which we are bringing somewhat up to date with the latest Linux releases. We have some security concerns about LILO and SILO. Several of these machines' consoles are openly available to the students, and we have been worried about the students forcing a reboot and bringing up Linux in single-user mode, gaining total access to the system. Admittedly, not much damage can be done from most of the machines, as most simply map their drives to the user directory of a more secure machine, but it's still a concern.

We've added the “password=/password/” and “restricted” lines to the respective /etc/lilo.conf and /etc/silo.conf files on each machine (and naturally added password protection to the BIOS to not allow booting from floppy); however, both lilo.conf and silo.conf are still readable to the average user. We want to retain the single-user mode availability for the lab crew and keep items contained in these config files, such as the image locations, available to those lab crew members without the security to modify the files. Is there a way to do this and yet prevent anyone from reading the password lines in lilo.conf/ silo.conf? Should we forego using LILO/SILO altogether and use something else? —Sam Hart, hart@physics.arizona.edu

Using another boot loader would be a good idea, but there's an alternative. The LILO configuration file is used ONLY when you actually run the LILO command. It's not required at boot time. Thus, you could put the file onto a floppy that only your lab crew has access to. They can mount it when necessary and use the **-C** option to specify its location when updating a kernel or changing a boot option. —Chad Robinson, chadr@brt.com

I would get the source code from SILO and LILO and hard-code the password in there (make sure, then, that the binary is a 700, so that a user cannot run strings on them). —Marc Merlin, merlin@varesearch.com

### Samba Problem

I have installed Samba 1.9.15p8 and I couldn't write to the Win98 PC. I am running Slackware 2.0.36. From the Linux box, I could read/write to Win98 box. From the Win98 box, I could read from my Linux box but not write. Any ideas?

```
(my /etc/smb.conf)
[public]
   path = /
   public = yes
   only guest = no
   writable = yes
   printable = no
```

—Hoo Kok Mun, hkmun@pacific.net.sg

Maybe you have some problems with Linux access rights on /. If you want to set up a public read/write directory, you should use a public directory such as /tmp. Here is my smb.conf config for /tmp:

```
[Tmp]
   comment = Temporary partition (rw)
   path = /tmp
   read only = no
   guest ok = yes
   case sensitive = no
   mangle case = yes
   preserve case = yes
```

If you want to test SAMBA more heavily from Win98, you may need to configure user access and passwords. Just add the following section in smb.conf:

```
[homes]
   guest ok = no
   read only = no
```

Don't forget Win98 uses encrypted passwords, so you should add the following lines in the [global] section of your smb.conf:

```
[global]
   security = share
   encrypt passwords = yes
   smb passwd file = /etc/smbpasswd
```

and add users and passwords with the **smbpasswd** command. —Pierre Ficheux, pficheux@com1.fr

Archive Index Issue Table of Contents

Advanced search

# New Products

**Ellen M. Dahl**

Issue #67, November 1999

ALPHAserver AT2000, UP2000 Motherboard, Cerebellum 1.3 and more.

## ALPHAserver AT2000

Atipa Linux Solutions announced its high-performance Linux server for enterprise applications, the ALPHAserver AT2000. When combined with dual 667MHz Alpha 21264 processors, the new server provides low-risk, high-availability, scalable solutions for increasing the performance of business-critical applications. Unlike a conventional server, the ALPHAserver AT2000 requires no configuration or reconfiguration of the workstations. The ALPHAserver AT2000 with one or two 667MHz Alpha 21264 microprocessors, up to 2GB of RAM and 6.4GB disk space starts at $5,999 US.

Contact: Atipa Linux Solutions, 2608-2 Main Street, Joplin, MO 64804, 800-360-4346, 417-626-2692 (fax), sales@atipa.com, http://www.atipa.com/.

## UP2000 Motherboard

Alpha Processor, Inc. (API) announced the UP2000, the first in API's Ultimate Performance Series motherboards that deliver the performance, bandwidth and reliability of an Alpha. The UP2000 is built on an extended ATX form factor and is a high-performance solution for scientific computing and Linux applications as a clustered server or workstation. Pricing for the motherboard with one Alpha Slot B starts at $4,554 US for 2MB of L2 cache. The UP2000 has a 667MHz Alpha Slot B processor, with a 750MHz Alpha Slot B processor available soon.

Contact: Alpha Processor Inc., 130C Baker Ave. Extension, Concord, MA 01742, 978-318-1100, 978-371-3177 (fax), info@alpha-processor.com, http://www.alpha-processor.com/.

### Cerebellum 1.3

Cerebellum Software, Inc. released version 1.3, an upgrade to Cerebellum enabling a wider variety of applications to easily access, integrate and update data located in an increased number of data source types. Major new features include access to mainframes and the Linux platform. Three new APIs have been added to support multiple development environments. A 30-day free trial developer's license can be downloaded from Cerebellum Software's web site. A developer's license can be purchased for $995 US. An enterprise package starts at $40,000 US.

Contact: Cerebellum Software, Inc., 600 Waterfront Drive, Pittsburgh, PA 15222-4716, 412-208-6500, 412-208-6521 (fax), info@cerebellumsoft.com, http://www.cerebellumsoft.com/.

### FusionX3

Western Scientific's FusionX3 is now available with Intel's Pentium-III 600MHz processor, making it the fastest non-Xeon powered workstation available for Linux and Windows applications. FusionX3s can be configured with dual 600MHz processors to give Red Hat Linux, Windows NT and Windows 98 users extraordinary performance. The FusionX3 is currently used for applications such as performance-demanding computer rendering, web serving and database management. Base price of a FusionX3 with a 600MHz processor is $3000 US.

Contact: Western Scientific, 9445 Farnham Street, San Diego, CA 92123, 858-565-6699, 858-565-6938 (fax), info@wsm.com, http://www.wsm.com/.

### The Best of Linux Distributions

Linux Press announced the second title in its new Linux Resource Series. *The Best of Linux Distributions* is based on Matt Welsh's cult classic *Linux Installation & Getting Started*, which has been revised, expanded and updated. Also included are four CD-ROMs containing the latest Linux operating system from Caldera OpenLinux, Debian GNU/Linux, Red Hat, and Slackware. The book is available for $39.95 US.

Contact: Linux Press, PO Box 220, Penngrove, CA 94951, 707-773-4916, 707-765-1431 (fax), sales@linuxpress.com, http://www.linuxpress.com/.

### Netwinder Office Server

Rebel.com, Inc. unveiled the NetWinder Office Server, providing small and medium-sized enterprises with full Internet and intranet network support while offering interoperability, compatibility, ease of use and performance. Based on the StrongARM RISC microprocessor and the Linux operating system, the NetWinder Office Server is configured with a broad range of network services and arrives ready for collaboration and communication. Pricing ranges from $895 to $3,850 US depending on the amount of RAM, hard-drive size and plastic or rackmount housing.

Contact: Rebel.com, 150 Isabella Street, 10th floor, Ottawa, ON K1S 5R3, Canada, 877-282-6735 (toll free), 613-230-8300 (fax), sales@rebel.com, http://www.rebel.com/.

### OpenWebConnect


OpenOrders released its newest product, OpenWebConnect. It allows any external TCP/IP-enabled device to easily connect to an OpenOrders database for order processing or retrieval of customer service information. OpenWebConnect allows Internet merchants to fulfill all of their order processing needs by providing a real-time link to OpenCatalog, a complete fulfillment and inventory management system. OpenOrders' unique scalable and open-architecture systems operate on servers running Linux as well as other platforms. Contact OpenOrders for pricing.

Contact: OpenOrders, Inc., 400 Centre Street, Newton, MA 02458, 617-527-5757 x22, 617-558-1361 (fax), sales@openorders.com, http://www.openorders.com/.

### RackMount-2UAXi

Rave Computer Association, Inc. has positioned its Rave Systems RackMount-2UAXi as an UltraSPARC-IIi server appliance operating on the Linux or Solaris operating system. It comes standard in a 19-inch 2U form factor chassis integrated with Sun's UltraAXi motherboard, a 300-watt auto-ranging power supply, six cooling fans and two 33MHz/32Bit PCI slots, and can be pre-loaded with Red Hat Linux 6.0 or Solaris 7. Contact Rave for price quotes.

Contact: Rave Computer Association, Inc., 36960 Metro Court, Sterling Heights, MI 48312, 800-966-7283, 810-939-7431 (fax), sales@rave.com, http://www.rave.net/.

### Red Hat Linux E-Commerce Server

Red Hat unveiled its Linux e-commerce server. This e-business solution combines the stability and scalability of open-source solutions, Red Hat Linux 6.0 and the Apache web server with all the security (RSA Data Security's 128-bit encryption engine) and e-commerce tools necessary to use Red Hat Linux in e-business applications. In addition, the included Linux applications CD contains Red Hat Linux 6.0 programs and a separate e-commerce directory. The package retails for $149.99 US. Corporations and ISPs can purchase additional support packages from Red Hat's new Response Link support center.

Contact: Red Hat Software, P.O. Box 13588, Research Triangle Park, NC 27709, 919-547-0012, 919-547-0024 (fax), orders@redhat.com, http://www.redhat.com/.

### Site Watch 2000

Internet technology firm NET Resolve launched Site Watch 2000 as their first major product offering. Site Watch 2000 is a combination of systems and network monitoring, reporting and 24x7 support for businesses that maintain Internet-based servers and networks. It runs on Linux and other operating systems. Prices are per server, ranging from $500 to $1000 US for setup, $275

to $1550 US monthly (Basic, Deluxe and Premierservice levels) with discounts on multiple server configurations.

Contact: NET Resolve, 378 Vintage Park Drive, Foster City, CA 94404, 877-638-7376 (toll-free), info@netresolve.com, http://www.netresolve.com/.

### Slam v1.2

Stabie-Soft announced the release of Slam v1.2, a family of integrated circuit layout editor tools. The tool suite includes a mask layout viewer (Slam-View), editor, extractor and delay integrator. All of the tools use the Tcl interpreter and provide programmatic access to the layout database. Supported platforms include Linux x86-based platforms using the 2.0 or 2.2 kernel. Price per node is $39 US for Slam-View; $3689 US per node for the full suite including Slam-View, Slam-Edit, XTK and Delay-Pak. A 30-day evaluation is free.

Contact: Stabie-Soft, 5828 Gentle Breeze Terr., Austin, TX 78731, 512-656-4713, sales@stabie-soft.com, http://www.stabie-soft.com/.

### CC_AUTH

Tri-Century Resource Group, Inc. announced the availability of CC_AUTH: Credit Card Authorization for Linux/UNIX and Java. Using CC_AUTH, any Java developer can integrate credit-card processing into an Internet or internal application within minutes. CC_AUTH is designed to connect with Authorize.net, a competitively priced credit-card authorization provider. A free 30-day CC_AUTH demo is available via Tri-Century's web site. Contact Tri-Century for pricing.

Contact: Tri-Century Resource Group, Inc., HC 79 Box 303, Wideman, AR 72585, 888-874-2368, 870-258-3168 (fax), http://www.tri-century.com/.

### TurboCluster Server

TurboLinux released TurboCluster Server, the industry's first high-availability clustering software solution for enterprise and the first clustering application available and scalable for web servers on Intel and Alpha architecture platforms. TurboCluster Server automatically detects failure in hardware, operating system, web server or router software and switches traffic around the damaged machine. Administration tools let users dynamically add cluster nodes and balance loads among the different servers in the cluster. TurboCluster Server has a suggested list price of $995 US for two nodes and $1995 US for three or more nodes.

Contact: TurboLinux, 2000 Sierra Point Parkway, Suite 702, Brisbane, CA 94005, 650-244-7777, 650-244-7766 (fax), orders@pht.com, http://www.turbolinux.com/.

### WebNow!

Screenshot


Unify Corporation introduced WebNow!, a dynamic web-page engine

designed to allow businesses to provide web access to their existing database applications via the Internet, intranet and extranet. It enables businesses to publish their database information on the Web securely, flexibly and without the need for complex programming. Unify WebNow! supports Linux and other leading server platforms, as well as all major databases. The WebNow! Development Kit is $8,000 US per developer, which includes Unify's VISION AppServer Enterprise Server. WebNow! Quick Start is $13,000 US, which includes one development kit and five days of consulting.

Contact: Unify Corporation, 100 Century Center Court, Suite 302, San Jose, CA 95112, 408-451-2000, 408-451-2007 (fax), info@unify.com, http://www.unify.com/.

### VERITAS Backup Exec and NetBackup

VERITAS Software announced its Linux backup support initiative to provide a storage management solution for the growing Linux user base. With VERITAS Backup Exec and VERITAS NetBackup, Linux servers and workstations are fully protected. VERITAS plans to support Caldera OpenLinux and TurboLinux this year. VERITAS Backup Exec for NetWare currently supports the Red Hat and SuSE distributions. VERITAS NetBackup provides advanced, database-aware solutions for all leading databases such as Oracle, DB2, Informix, Sybase and Microsoft SQL Server. VERITAS Backup Exec for NetWare has a U.S. suggested retail price of $1,395 (multi-server version). Linux support pricing starts at $200 with VERITAS NetBackup.

Contact: VERITAS Software Corporation, 1600 Plymouth Street, Mountain View, CA 94043, 650-335-8000, 650-335-8050 (fax), vx-sales@veritas.com, http://www.veritas.com/.

Archive Index Issue Table of Contents

Advanced search

# NoSQL Tutorial

**Giuseppe Paternó**

Issue #67, November 1999

A comprehensive look at the NoSQL database.

Some months ago I had a discussion with NoSQL creator, Carlo Strozzi, regarding the databases.

I should admit, I am an SQL fan! It's hot having the same language, no matter which platform or database engine is used. He underlined the fact that most SQL engines lack of flexibility and waste system resources (memory and disk space) because of their multi-platform environment (such as Oracle, DB2, Informix, etc.).

He suggested I have a look at the white paper that inspired him: “The UNIX Shell As a Fourth Generation Language” by Evan Schaffer (evan@rsw.com) and Mike Wolf (wolf@hyperion.com).

Quoting from the above paper:

> ... almost all [database systems] are software prisons that you must get into and leave the power of UNIX behind. [...] The resulting database systems are large, complex programs which degrade total system performance, especially when they are run in a multi-user environment. [...] UNIX provides hundreds of programs that can be piped together to easily perform almost any function imaginable. Nothing comes close to providing the functions that come standard with UNIX.

The UNIX file structure is the fastest and most readily-available database engine ever built: directories may be viewed as catalogs and tables as plain ASCII files. Commands are common UNIX utilities, such as **grep**, **sed** and **awk**. Nothing should be reinvented.

NoSQL was born with these ideas in mind: getting the most from the UNIX system, using some commands that glue together various standard tools. Although NoSQL is a good database system, this is not a panacea for all you problems. If you have to deal with a 10 gigabytes table that must be updated each second from various clients, NoSQL doesn't work for you since it lacks of performance on very big tables, and on frequent updates you must be in real time. For this case, I suggest you use a stronger solution based on Oracle, DB2 or such packages on a Linux cluster, AS/400 or mainframes.

However, if you have a web site containing much information and more reading occurs than writing, you will be surprised how fast is it. NoSQL (pronounced noseequel, as the author suggests) derives most of its code from the RDB database developed at RAND Organization, but more commands have been built in order to accomplish more tasks.

## Installing NoSQL

The latest NoSQL source code can be found at ftp://ftp.linux.it/pub/database/NoSQL, but RPM and Debian packages are also available. At the time of writing, latest stable version is 2.1.3.

Just unpack the source using the command **tar -xzvvf nosql-2.1.3.tar.gz** in a convenient directory (such as $HOME/src), and you will get all the code in the nosql-2.1.3 subdirectory. Enter the above subdirectory and do the following:

```
./configure
make
make install
```

The software will put its engine into /usr/local/lib/nosql, its documentation in /usr/local/doc/nosql and a symlink /usr/local/bin/nosql that points to the real executable (/usr/local/lib/nosql/sh/nosql). You can change the directory prefix (e.g., /usr instead of /usr/local) invoking **./configure --prefix=/usr**.

You should copy the sample configuration file to the $NSQLIB directory (i.e., /usr/local/lib/nosql). This is not required but it's useful for changing some parameters via configuration file instead of variables. The commands

```
cp nosql.conf.sample /usr/local/lib/nosql/nosql.conf
chmod 0664 /usr/local/lib/nosql/nosql.conf
```

will copy it with the right permissions. You can optionally have a personal NoSQL configuration file creating an $HOME/.nosql.conf with 0664 permission applied.

Although NoSQL installation is quite simple, I suggest you to read the INSTALL file: the author gives some good tips.

## Getting Some Steps Around

Now that the package has been installed, let's start getting acquainted with NoSQL commands through an example.

We are the usual Acme Tools Inc. which supplies stuffs to the Toonies land. We would like to track our customers, so we should create a first table in which we will list some customers details (such as code, phone, fax, e-mail, etc...). The best way to create tables from scratch is via template files.

A template file contains the column names of the table and associated optional comments separated by tabs and/or spaces. Comments may be also specified with the usual hash sign (#) as the first character of the line. The file below, customer.tpl, is our template for the customer table.

```
# Acme Tools, Inc.
# Customers table
#######################
CODE    Code number
NAME    Name/Surname
PHONE   Phone no
EMAIL   E-mail
```

Most of the NoSQL commands read from STDIN and write to STDOUT. The **maketable** command, which builds tables from templates, is one of these. Issuing the command

```
nosql maketable < customer.tpl
```

we'll get the table header on STDOUT:

```
CODE    NAME    PHONE   EMAIL
----    ----    -----   -----
```

Great, but we should keep it in a file. We could simply redirect the command output to a file, e.g.,

```
nosql maketable < customer.tpl > customer.rdb
```

but this wouldn't be the right way. The **write** command may be helpful in this case, as it reads a table from STDIN (a simple header in this case) and writes a file, checking data integrity.

The resulting command would be

```
nosql maketable < customer.tpl |
  nosql write -s customer.rdb.
```

The **-s** switch in the write operator suppress STDOUT, e.g., **nosql write**, similar to the **tee** UNIX utility, writes both to file and STDOUT, unless **-s** is specified.

Pay attention, because the write command does not do any locking on the output table: **nosql lock table** and **nosql unlock table** must be used for this purpose.

### First Editing: Using edit Command

Now let's add our first customer with the command

```
nosql edit customer.rdb
```

The default editor is the **vi** command, but you can use your favorite editor changing the EDITOR environment variable. The screen below is presented to the user:

```
CODE
NAME
PHONE
EMAIL
```

Just fill the fields with some information, remembering to separate values from field names with a tab. *Do not delete* first and last blank lines, this is not a bug: it's the way NoSQL handle lists. But I prefer to let you discover this little feature later in this article.

```
CODE    ACM001
NAME    Bugs Bunny
PHONE   1
EMAIL   bugs.bunny@looneys.com
```

Now that we have filled in the form, just write it (**ESC** then **:wq!**) and the command will check if the format is correct and write it to disk. Wow, we have a real table and real data!

### NoSQL Table Format

Since we are curious, we will taking a look to the real file on disk.

```
CODE    NAME    PHONE   EMAIL
ACM001  Bugs Bunny      1
```

First of all, is important to note the fact that all columns are tab-separated: please keep this in mind when you want some external program to update the table, otherwise you will break the table integrity.

The first line is called the *headline* and contains column names; the second is the *dashline* and separates the headline from the body: both are named the *table header*. The rest is called the *table body* and contains the actual data.

A number of commands have been build to displays these parts, and they are simply calls to ordinary UNIX utilities:

- **nosql body**: displays the table body (same as: **tail +3 > table**)
- **nosql dashline**: displays the table dash line (same as: **sed -n 2p < table**)
- **nosql header**: displays the full table header (same as: **head -2 < table**)
- **nosql headline**: displays the table headline (same as: **head -1 < table**)
- **nosql see**: displays the **TAB** character as **^I** and **newline** as **$**, making much easier to see what's wrong on a broken table (same as: **cat -vte < table**)

Once again, this shows how powerful the UNIX OS is on its own, and how handy it can be for add-on packages such as NoSQL to tap into this power without having to re-invent the wheel.

### Simple Data Insertion: Using Shell Variables

A fun way to fill the table is using the environment variables. You can export variables in any way, e.g., using UNCGI in a CGI environment or named as column names with the desirable values as follows:

```
export CODE="ACM002"
export NAME="Daffy Duck"
export PHONE="1-800-COOK-ME"
export EMAIL="dduck@looneys.com"
```

Then issue the command:

```
nosql lock customer.rdb; env | nosql shelltotable |\
nosql column CODE NAME PHONE EMAIL |\
nosql merge CODE NAME PHONE EMAIL customer.rdb |\
nosql write -s customer.rdb; nosql unlock customer.rdb
```

and the work is done—a bit cryptic? Yes, but that's the power of NoSQL: all can be done in a single shell command. Let's explain it:

- **nosql lock customer.rdb**: this locks the table and ensures noone else can write in the table at the same time we do.
- **env**: prints the environment variable.
- **nosql shelltotable**: reads all variables from the pipe and writes a single record table containing all values to STDOUT.
- **nosql column CODE NAME PHONE EMAIL**: reads the NoSQL table containing the environment variables from the pipe and selects column CODE, NAME, PHONE and EMAIL in that order and writes STDOUT.
- **nosql merge CODE NAME PHONE EMAIL customer.rdb**: reads the two merging tables, one from pipe (STDIN) and other from file, writing the merged table to stdout. The resulting table has two records: the existing one and the new one extracted from the above process.

- **nosql write -s customer.rdb**: reads the resulting table (merged from the above command) and writes it to disk as customer.rdb. We already explained what switch **-s** means.
- **nosql unlock customer.rdb**: unlocks the table.

Now have a look to the resulting table using the command that reads the table: **nosql cat customer.rdb**.

```
CODE    NAME              PHONE           EMAIL
------  ----------------  --------------- ---------------------
ACM001  Bugs Bunny        1-800-CATCH-ME  bugs.bunny@looneys.com
ACM002  Daffy Duck        1-800-COOK-ME   dduck@looneys.com
```

### More Insertion: The Lists

NoSQL can handle data in different way, called list format. A sample table may be:

```
CODE    ACM003
NAME    Bart Simpson
PHONE   1-555-5432-321
EMAIL   bart@springfield.org
CODE    ACM004
NAME    Wiley The Coyote"
PHONE   1-800-ILLGETIT
EMAIL   wiley@looneys.com
```

Yes, you're right! This is the same way the edit command displays data. Although list tables aren't performing at all, in my opinion they are a good way to insert new data into tables. It's handy creating a program that can output this fashion. An example is shown in Listing 1.

Okay, this is not a “state of the art” shell program, but this example may show that the complete operation is easy with *every* language, even the shell.

There are a couple of things I would like to emphasize about the above code. How can a list be merged with a real table? The command **listtotable** does the job for you, as you probably guessed looking at the pipe, converting the list format to the table one. A reverse command, **tabletolist**, exists as well.

Please notice the beginning and ending newlines, as well as the tabs between field name and value, which appear in the print statement: these are required in order to create the correct list structure.

The list structure, as well as table structure, are well documented in the Chapter 2 of the NoSQL reference you will find in /usr/local/doc/nosql/doc.

### Getting More from NoSQL

Now, let's have a business example? A complete catalog was created using a NoSQL table (see the catalog.rdb file below) and published on the Web

dynamically. Every two days, we receive orders from our customers that they have usually created with Excel and exported, at our request, in a coma-separated file.

```
PRID    DESC                            PRICE
------  ------------------------       ------
PRD001  Acme glue for RoadRunners        30.00
PRD002  Acme TNT                        150.00
PRD003  Carrots                           5.00
PRD004  Acme toolbox                     75.00
The file (sample_order.txt below) we receive has the following
format: requester's unique code, Product ID, Quantity.
```

```
ACM004,PRD001,5
ACM004,PRD002,30
ACM004,PRD004,1
```

Now from a shell or command line we run:

```
export TMPFILE=`mktemp -q /tmp/$0.XXXXXX` ; cat sample_order.txt | \
perl -e 'print "CODE\tPROD\tQTY\n"; print "----\t----\t---\n"; \
while(
catalog.rdb | \
nosql addcol SUBTOTAL | nosql compute 'SUBTOTAL = QTY*PRICE' > $TMPFILE ; \
echo "Please bill to:" ; echo "---------------" ; echo ""; cat
$TMPFILE |\
nosql join -j CODE - customer.rdb | nosql column NAME PHONE EMAIL | \
nosql body | head -1 ; echo "";echo "" ; cat $TMPFILE | nosql rmcol CODE | \
nosql print -w; echo ""; echo -n "Total due: "; cat $TMPFILE | \
nosql subtotal -T SUBTOTAL | nosql body ; rm $TMPFILE
```

and our output is:

```
Please bill to:
---------------
Wiley The Coyote        1-800-ILLGETIT  wiley@looneys.com
PROD   QTY DESC                       PRICE SUBTOTAL
------ --- ------------------------- ------ --------
PRD001   5 Acme glue for RoadRunners  30.00      150
PRD002  30 Acme TNT                  150.00     4500
PRD004   1 Acme toolbox               75.00       75
Total due: 4725
```

This result may be sent via e-mail to our logistic people. Not so bad for a *five-minute single shell command*, is it?

I know it's a bit hermetic, so let's have a closer look: the explanation is divided in four sections to be easily read. I will keep out the **echo** commands which are quite obvious.

### Section 1: Extracting Useful Data From the Received File

- **export TMPFILE=`mktemp -q /tmp/$0.XXXXXX`**: exporting environment variable **TMPFILE** that contains a runtime generated tempfile.
- **cat sample_order.txt**: get as input the file we received.
- **perl -e 'print "CODE\tPROD\tQTY\n"; print "-----\t----\t---\n"; while(<STDIN>) { s/,/\t/g; print };`**: prints a NoSQL compliant header, composed of CODE,

PROD and QTY. Then, it prints the table received from STDIN, substituting a coma (,) with a tab (\t), in order to get the correct table structure.

- **nosql join -j PROD - catalog.rdb**: joins the table read from STDIN (the **-** character) and catalog.rdb using the PROD (product ID) column.
- **nosql addcol SUBTOTAL**: now a column SUBTOTAL is added to the resulting table.
- **nosql compute 'SUBTOTAL = QTY*PRICE' > $TMPFILE**: calculates the SUBTOTAL column by multiplying quantity and price columns. It then redirects the resulting table to the previous calculated temporary file.

### Section 2: Getting and Printing Billing Name

**cat $TMPFILE**: reads the table written from previously stored temp file.

- **nosql join -j CODE - customer.rdb**: joins the STDIN table (minus sign) and the customer.rdb table on the CODE column.
- **nosql column NAME PHONE EMAIL**: selecting NAME, PHONE and EMAIL columns.
- **nosql body**: gets only table content, without printing the header lines.
- **head -1**: prints only one line; all the lines are identical since who sent the file (or order) is the same customer.

### Section 3: Printing the Order Content

- **cat $TMPFILE**: reads the table written from previously stored temp file. "**nosql rmcol CODE**: removes CODE column from the STDIN table (it's not useful for us at this moment).
- **nosql print -w**: prints the results in a simple but useful form. Columns containing only numbers are right-justified with blanks, while anything else is left-justified. The **-w** switch forces the **print** command to fit in a terminal window.

### Section 4: Getting Total Amount Due

- **cat $TMPFILE**: reads the table written from previously stored temp file.
- **nosql subtotal -T SUBTOTAL**: calculates the sum of the SUBTOTAL column. The result of this command is a NoSQL compliant table.
- **nosql body**: gets only table content, without printing the header lines.
- **rm $TMPFILE**: removes the temporary file created before. The commands used at this moment have, of course, more options. For a complete overview, please check out the documentation included with the NoSQL package.

### Indexing the Tables

The best way to search information on a large amount of data is, of course, indices, and NoSQL obviously has its own operators to interact with them.

Suppose our customers are increasing greatly since Acme Inc. started, so we need to speed up our searches. Most of the time, we look up on the CODE column, so it would be better to build the index on it last.

The command **nosql index customer.rdb CODE** will create an index on customer.rdb table against the CODE column. Index files are named by appending an x and the column name(s) (separated by a dot) to the base name of the table it refers to. In our case, the index file name is customer.x.CODE. If we want to update the index without rebuilding it, we should run

```
nosql index -update customer.x.CODE
```

A crontab job on our system will ensure a periodic index update, by running the command:

```
cd /var/tables/acmeinc; nosql index -update
```

Not specifying any index file name on **nosql index -update** will update all indices in the current working directory.

Now that indices are built, let's try a search. In order to search against indices, you should create a small key table as input to the **nosql search** command, for example:

```
echo -e "CODE\n----\nACM004" | nosql search -ind customer.x.CODE
```

This command will extract the **ACM004** value in the CODE column using the customer.x.CODE index. Creating this key table may not seem handy, but this ensures the maximum reuse of commands. Suppose you extracted some data from table1, you can search this last result on table2 easily with a single pipe. Try to think a bit about it, this is not a bad idea at all.

### Your Lifebelt: The RCS

What if you discovered that last write caused a disaster and you do not have a backup? You will never have the latest copy of the table before the disaster happened.

The Revision Control System (RCS) is one of the best configuration management tools available, it can be used for version control of many types of files, including tables. The command **nosql edit**, for example, will automatically

check out a table for editing, and then check the new version back into RCS. Other commands can utilize tables that are under RCS control by using explicit commands like:

```
co  -p  table | nosql row  ... | nosql column  ... | nosql print
```

or relying upon the **cat** command to handle interactions with RCS automatically:

```
nosql cat table | nosql row  ... | nosql column  ... | nosql print
```

Note that this checks out a table, sends it to **nosql row**, then to **nosql column**, and finally, prints the data with **nosql print**. In general, any series of commands necessary can be constructed to do a given task even if the tables are checked into RCS.

Now that you have RCS keeping watch for you, if the disaster happens in real life, you can easily **rollback**: the command

```
nosql rollback [-d datestring] tablename
```

will extract the table updated at the desired time.

### Security

NoSQL works with UNIX, not in addition to it. Set correct permissions to files using groups: create a group for those users who can access these files. This is a great security wall.

### Performance Tips

No matter what program you have to deal with, sooner or later you will have to deal with performance. I'm not a developer of the NoSQL Database System, but I can give you some useful advice.

First of all, keep your tables small. Don't keep all the data in a single table, this is a waste of performance. Using the method that best suits your environment, try splitting tables into several files and organizing them into directories. In our examples, we can keep track of our customers creating a single “phone directory” (i.e., customer.rdb), then creating a directory for each order status (received, waiting_for_bill, archive), and last a table for each customer (the file name will be the customer code). For example:

```
/var/tables/acmeinc/
customer.rdb
                    catalog.rdb
                    received/
                             ACM001.rdb
                             ACM003.rdb
                    w4bill/
ACM002.rdb
```

```
                              ACM003.rdb
                    archive/
                              ACM001.rdb
                              ACM002.rdb
                              ACM003.rdb
```

If you must do everything in a big table and you have to update it frequently, there's a trick for you if it's an indexed file: journaling.

Create a journaling table, say customer.j, with exactly the same header as customer.rdb, but containing only those records which we want to insert into, remove from or append to customer.rdb. The entries in customer.j must be in a format suitable for the **nosql merge** command.

Whenever we fetch data from customer.rdb we will have to do it in three logical steps. The first step is to use **search** on bigtable to take advantage of the indices. This will produce an intermediate output that will then be merged into customer.j, and the final output will undergo the original query statement again.

Any updates to customer.rdb will be done to customer.j with the syntax described in the documentation of the **merge** command (there you'll find how to use the operator to the optimum level). You will also have to make sure that customer.j is kept sorted on its primary key field after each update. For example, if you have an index on the CODE column in the customer.rdb table, you should use:

```
echo -e "CODE\n----\nACM004" | nosql search -ind customer.x.CODE |\
nosql merge CODE NAME PHONE EMAIL customer.j | \
nosql row 'CODE=="ACM004"'
```

As you can see, the trick is:

1. Perform an indexed search on customer.rdb to quickly obtain a much smaller (possibly empty) subset of the table.
2. Merge the first output with customer.j on the fly during the query.
3. Do a sequential post-query on the final output.

If that's not enough, another trick to improve speed is to make your AWK use the **ash** (by Kenneth Almquis) for **system()**'s and pipes, since this shell is small and fast at startup. Suppose your AWK is /usr/bin/mawk and your shell is /bin/ash (a fairly common case, especially on Debian GNU/Linux), then you can do something like this:

1. Create the following hard link: **ln /bin/ash /bin/ah**
2. Modify AWK to make it use /bin/ah as opposed to /bin/sh, and write the modified AWK to /usr/local/bin/nsq-mawk: sed 's/\/bin\/sh/\/bin\/ah/g' /

usr/bin/mawk\> /usr/local/bin/nsq-mawkchmod 755 /usr/local/bin/nsq-mawk

3. Modify the NSQAWK value in the config file (/usr/local/lib/nosql/nosql.conf or $HOME/.nosql.conf) to /usr/local/bin/nsq-mawk. This will speed up your queries !

### Going to the Web

Using NoSQL on the Web is a matter of seconds. Let's suppose that Acme Tools Inc. now has a web site, and you want your customer to search on database for their pending orders. We first create a small input form, the file getname.html show in Listing 2, on which we ask for customer name. I did not use any security at all in this example, but at least password should be asked in a production environment.

Since we are not good at graphics, we create a small template (result.html) that can be modified easily. This template, result.html, is shown in Listing 3.

In this template, some keywords are substituted by the CGI: our standard keywords start and end with a double hash (#) sign, for example ##KEYWORD##.

A special section of the template, named stanza, will be repeated as many times as the number of query rows. The stanza starts and ends with special comments that will be recognized by the CGI.

Now is the time for writing the CGI: Listing 4, result.pl, is a perl script that should perform the queries based on the input name and then creates a resulting page based on the previous template. Most of the queries used in this CGI are those used in previous examples, so we won't repeat them. Just have a look to the main query:

```
@cusdata = `nosql cat $datafile | nosql join -j PROD - $ctlgfile |
nosql addcol SUBTOTAL | nosql compute 'SUBTOTAL = QTY*PRICE' | nosql
column PROD DESC QTY PRICE SUBTOTAL | nosql body`;
```

The thing to notice is the array that will contain query rows: each row contains a tab separated list of fields, as the NoSQL table row specification. In fact, in the stanza keyword substitution, a **split** function against tab is used:

```
my ($prod, $desc, $qty, $price, $subtotal) = split(/\t/, $data);
```

All queries run with back ticks that, instead of the **exec()** and **system()**, return the STDOUT of the program. This output may be reused in the program using variables. A negative fact of this is security: you cannot run this program in tainted mode (**-T** switch in the perl command line), but this can be avoided with

some tricks such as the ones I used. First of all, you should avoid buffer overruns by using a substring function **($cusname = substr($cusname, 0, 50))**, then keeping out some escape characters (such as > Once the queries have been executed and we have all the necessary values, we load the template file and associate it with the default input and pattern-searching space. The template file, in which we transformed new lines and multiple spaces into a single space, is now divided into three parts using pattern matching: the header, the body (aka stanza) and the footer.

```
/(.+)<\s*!--\s*here\s+starts\s+nosql\s+stanza\s*--->(.+)<\s*!--\s*here
\s+ends\s+nosql\s+stanza\s*--\s*>(.+)/i ;
```

This search will identify the stanza into the template, using the keyword **<!-- here starts nosql stanza -->** as the beginning and **<--! here ends nosql stanza -->** as the end. As you can notice, these are simple HTML comments, so can be introduced easily by our graphic experts. All items before the beginning comment is considered the header, while the rest is the footer.

Before entering the keyword processing in the body, we will do it in the header and footer in order to set proper customer name and total amount due:

```
$header =~ s/##NAME##/$cusname/;
$footer =~ s/##TOTAL##/$total/;
```

The final part is the keyword substitution in the stanza. Here we will swap the original variable (*$body*) with a temporary one (*$tmpbody*), in order to leave the first unchanged for next loop. Here the fields are split using the method I described earlier, then substituted for the keywords in the template file. Of course, there are thousands of way of writing down this kind of CGI, be it in Perl or other languages. Write one in your favorite language and let your imagination be your guide: databases are plain ASCII files, so you can process them as you like, and you will get great results.

For a REAL example of NoSQL usage on the web, check out http://www.whoswho-sutter.com/, http://annunci-auto.repubblica.it/ and http://www.secondamano.it/ (the first is in English, while the others are in Italian).

## NoSQL in the Near Future

Talking to Mr. Strozzi, the main developer of NoSQL, he revealed to me some news on ongoing development of the RDBMS.

The first minor modification you will see at a glance is the version number: kernel release schema has been introduced, so even numbers are the stable ones, while odd ones are unstable (current unstable is 2.3.1).

The major modification is the **rel** command. It checks table reference integrity before an update/insert/delete, but won't take any action: it only advises you if something will be broken , so you should use it in your program before you do any table operation.

Other minors enhancements are some commands such as **insert**, **delete**, **tabletoperl**, **perlencode** and **tabletom4** that are quite useful in a programming environment as well as on the command line. At present, no official reference for those commands, but comments in the source code will easily let you understand how to use them.

Mr. Carlo Strozzi told me that the next stable release, 2.4.0, will be available around November of 1999.

## Conclusion

NoSQL is a great database system for web-based applications, in which reading occurs much more than writing. I recommend it also for full-text searching and in those applications where ASCII tables may be handy.

For more information have a look at the official web page http://www.mi.linux.it/People/carlos/nosql/ or subscribe to the mailing list by sending a message to noseequel@mi.linux.it with the words **subscribe noseequel** in the **Subject:** line of the message.

I would like to thank the NoSQL creator, Carlo Strozzi, for being supportive of me in writing this article; Maurizio Sartori, who gave me some hints; Giovanni Granata, Andrea Bortolan and all the people who have encouraged me to go on researching.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue67/3294.tgz.

**Giuseppe Paternó** (gpaterno@spacelab.gpaterno.com) has recently changed jobs from System Engineer for IBM Italy, where he worked on firewalls, Lotus Domino, AS/400 and mail systems, to Internet System Administrator for Infostrada, a local Telco/ISP. He likes music, cooking, science, and of course, beer.

LINUX JOURNAL



# Interfacing Relational Databases to the Web

**Will Benton**

Issue #67, November 1999

This document explains how to build a database-backed web site using Apache, the PHP3 module and the PostgreSQL relational database.

Why would one want to link a database to the web? A better question might be, “Why *wouldn't* one want to link a database to the web?” Static HTML pages are good for many things: documentation, hypertext books, personal pages and other unchanging information. However, static pages present a few problems:

- Static pages can be a hassle to maintain. If one is managing a large site with thousands of pages, changing just the “look and feel” of the site will involve either an inordinate amount of work or a long evening with CGI and Perl. This gets even nastier when the content of the site changes.
- Static pages don't allow for user input, feedback or collaboration. Suppose you want to add a message board to your static web site. You can set up a form that mails user comments to the webmaster, who manually puts them on a page—you can even set up a script to do this. However, this presents a few problems, as we shall discuss later.
- Static pages don't allow you to operate a web service. One can operate a web site with static HTML, but some of the most useful web sites today, such as Slashdot (http://slashdot.org/), CNN (http://www.cnn.com/) and Amazon.com (http://www.amazon.com/), are services offering dynamic database-backed content.

A database management system provides uniform access to structured data, much like an object-oriented programming language provides a uniform way to store data and define methods to act on that data. If you're writing a program in C++ and you need to deal with strings, would you rather wrap all associated routines in a class or deal with the fact that your string is really a **char *** each time you operate on it? With a database, one is offered a layer of abstraction between how the data is actually stored and how one uses it, which is quite

useful for several reasons. You could achieve the same results with C or Perl scripts, but it wouldn't be as pretty and could get downright ugly.

I hope this quick rundown is enough to convince you that you really want to use a database for your dynamic web site. In this article, I will present instructions for installing PHP3 (http://www.php.net/) and PostgreSQL (http://www.postgresql.org/), a little bit of theory, some instructions for using SQL and PHP3, and an example application.

## Installing the Tools

This description assumes you are running Red Hat, but most of these instructions will be applicable to other distributions; these tools are fairly painless to install from source, anyway.

Here's a list of what you'll need to run the example application and develop your own applications:

- Apache 1.3 or greater. You will need at least version 1.3, because 1.2 does not support modules, and PHP is faster and more secure as a module. On a Red Hat system, you'll need both the **apache** and **apache-devel** packages. (Make sure you have the file apxs installed, because we're going to recompile the PHP3 Apache module.)
- The source RPM for PHP3, version 3.0 or greater.
- PostgreSQL 6.4 or greater; the example code will not run on version 6.3 without a little editing, because 6.4 has an SQL parser.

To rebuild PHP3 for PostgreSQL support, take the following steps:

1. Use **su** to become root.
2. Install the source RPM for PHP3 (**rpm -ihv mod_php3-3.0.5-2.src.rpm** on Red Hat 6.0). This will place a “spec file” in the directory /usr/src/redhat/SPECS and a tar file of the source in /usr/src/redhat/SOURCES. Since the PHP module that comes with Red Hat doesn't have database support enabled by default, we'll have to recompile it. RPM makes this fairly painless.
3. Because the PHP3 installation process assumes a default PostgreSQL installation, not the Red Hat one, we'll need to make some symlinks. Create a directory /usr/local/pgsql and make symbolic links from /usr/include/pgsql to /usr/local/pgsql/include and from /usr/lib/ to /usr/local/pgsql/lib.
4. Invoke your favorite editor on the spec file (mod_php3.spec) and search for ./configure; then add the configure option **--with-pgsql**.
5. Now build a binary package with rpm: **/rpm -bb mod_php3.spec/**

6. If all goes well, you'll have a binary package in /usr/src/redhat/RPMS/arch, where /arch is your architecture. Install it, and you're ready to move on.

### Setting up PostgreSQL

PostgreSQL uses a different access system than the rest of your system; oddly enough, not even root has access to the database by default. The database system has its own user system and passwords, and postgres is the database administrator account by default. The advantage to the separate access system is that one can create database users who do not have UNIX accounts; this way, the database for your web application can specify access control without creating a potential security hole for your system. To add your web administrator (web) as a database user, use **createuser** (as root):

```
# su postgres -c createuser
Enter name of user to add ---> web
Enter user's postgres ID, or RETURN to use UNIX user id: 542 -> 542
Is user "web" allowed to create databases (y/n) y
Is user "web" allowed to add users? (y/n) y
createuser: web was successfully added
```

Then, as web (or whatever account you used), you'll be able to create a database with **createdb foo** and then try some queries on foo using **psql foo**.

You'll also need to set up PostgreSQL to accept incoming TCP/IP connections so your PHP3 pages can access it. Fortunately, System V **init** makes this easy. Simply open the file /etc/rc.d/init.d/postgresql and change the line

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

so that it reads

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

While you're at it, you will probably want to specify a different port from the default (5432) for security reasons. To run the PostgreSQL back end on a different port, merely append a **-p *port*** to the above line.

### All the SQL You Need To Know (Not Really)

Just about every relational database in the world uses SQL (or some extended version of SQL) as its query language. SQL allows you to define tables, select records based on given criteria, update values in one or many records and delete records. This is just a brief introduction to SQL; for more complete references, see Resources.

## Creating Tables

To create a table, one uses the **CREATE TABLE** statement. Its syntax is as follows:

```
CREATE TABLE tablename (field-1 type-1, ..., field-n type-n)
```

In psql, you will need to end each statement with a semicolon. These semicolons are not part of the SQL language, but rather for the benefit of psql's lexer.

You may also declare fields as **NOT NULL**, **UNIQUE** or **PRIMARY KEY**, or specify a value as **DEFAULT** to a field. PostgreSQL will create an index on primary key fields. Unfortunately, as of version 6.4, PostgreSQL does not support foreign keys, but at least the parser will not choke on the SQL **REFERENCES** keyword.

Here's an example, akin to the UNIX password file:

```
CREATE TABLE passwd
   (username    varchar(8) PRIMARY KEY,
   -- PRIMARY KEY implies UNIQUE
    cryptedpass char(13),
    uid         int UNIQUE NOT NULL,
    gid         int NOT NULL,
    gecos       varchar(80),
    -- the GECOS field (real name, office, etc.)
    homedir     varchar(80),
    shell       varchar(50) DEFAULT '/bin/sh');
```

Note that SQL uses a double-dash to begin comments, which are terminated by a newline.

## Inserting Data

To insert data into a table, use the **INSERT** statement:

```
INSERT INTO tablename (field-1, ..., field-n)
VALUES (value-1, ..., value-n/)
```

You needn't specify field names if you are inserting values into every field. Here's an example for the table we just created:

```
INSERT INTO passwd (username, cryptedpass,
   uid, gid, gecos, homedir, shell)
VALUES ('fred', '37MniLTaiPLaL', 42, 500,
   'Fred Mbogo', '/home/fred/', '/bin/sh');
```

Note that SQL uses single quotes for string constants. Any closet Pascal programmers will feel right at home.

### Retrieving Data

The SQL **SELECT** statement returns records where values meet a certain criteria. Here are some examples of **SELECT** in action:

```
SELECT * FROM passwd;
-- returns all fields of all records
SELECT username FROM passwd;
-- returns all usernames
SELECT * FROM passwd WHERE username = 'fred';
SELECT * FROM passwd ORDER BY username, shell;
SELECT * FROM passwd
     WHERE homedir LIKE '/home%'
     -- % is the SQL wildcard character
     AND   shell = '/bin/sh'
     ORDER BY username;
SELECT homedir, projectname
     FROM passwd, projects
     -- assuming we have a projects
     table
     WHERE
     -- this will return the home directory of
       passwd.username = projects.leader;
       -- each project leader for each project
```

### Keeping Your Data Current

To change field values in records, use **UPDATE**:

```
UPDATE tablename SET field-1 =
value-n WHERE qualification
```

The **WHERE** is optional, but if you don't specify a **WHERE** clause, SQL will update *all* the records, which is clearly the "Wrong Thing".

Let's say Fred Mbogo wants to change his shell. This script will do it:

```
UPDATE passwd SET shell = '/bin/tcsh' WHERE username = 'fred';
```

### Deleting Records

To delete records, simply use **DELETE**:

```
DELETE FROM tablename WHERE qualifier
```

Just like **UPDATE**, the **WHERE** is optional, but you probably want it anyway. Let's say Fred has offended his sysadmin one too many times:

```
DELETE FROM passwd WHERE username = 'fred';
```

### All the PHP3 You Need To Know (Not Really)

The on-line PHP3 manual, http://www.php.net/manual/, is an excellent reference and will be necessary reading before you create your own database

web application. Furthermore, it is a database-backed web site and has lots of user comments. Here, we will examine just the most basic PHP3 features.

### Using PHP3

Here is a simple PHP3 program, which demonstrates some basic features. Note the separate HTML and PHP3 blocks:

```
<title>Hello, world!</title>
<body>
<?php
  echo("Hello, world!\n");
  echo("<p>\nWhat a <b>bold</b> move this is!\n");
?>
</body>
```

This program will send the following HTML to the remote browser:

```
<title>Hello, world!</title>
<body>
Hello, world!
<p>
What a <b>bold</b> move this is!
</body>
```

A similar program, which takes an argument, would look like this:

```
<title>Hello, world!</title>
<body>
<?php
  echo("Hello, $name!\n");
  echo("<p>\nWhat a <b>bold</b> move this is!\n");
?>
</body>
```

You would view this page (assuming you called it hello.php3) like any CGI script: http://*yourhost*.net/~fred/hello.php3?name=fred. This, of course, assumes you are named Fred and have put this file in your /public_html directory.

### Database Connectivity

PHP3 provides a number of useful functions for connecting to databases; the best place to read up on these is www.php.net/manual/ref.pgsql.php3, and we shall examine a few of them.

```
int pg_connect(host, port, options, tty, dbname);
```

This function returns an integer, the “connection index”, which you will need for all operations on this connection. If a connection can't be established, it returns zero.

- **int pg_exec(*conn*, *query*);** Executes the SQL query *query* on connection *conn*. Returns a result set index.

- **int pg_numrows(*result*);** Returns the number of tuples in the result set *result*.
- **array pg_fetch_row(*result*,**Returns an array of values corresponding to the row *row* of result set *result*.
- **void pg_close(*conn*);** Closes the connection *conn*.

### Example Application: A Multiuser Address Book

Our example application is an address book that one can access over the Internet. A user logs in with her name and password and is presented with a menu of options, including browsing and searching the address book and adding a new person. For each person in the address book, the database stores an arbitrary number of e-mail addresses, telephone numbers, URLs and postal addresses. This address book also has some nifty features like mailing passwords to new accounts and automatic **mailto** and **href** links for e-mail and web addresses.

### Design Considerations

We have already completed the first step of the design process—deciding what we need our application to do. What remains is defining the data that our application will access in three iterations:

1. A high-level data model
2. A low-level data model
3. A set of views and transactions that are legal for application users

Generally, one initially models databases in an entity-relationship diagram, which represents each table as an *entity* with a set of *attributes*, and a "join", or query, spanning multiple tables is represented by a *relationship*. Even if you're not going to go to the trouble of making up an E-R diagram, you should at least consider these concepts; we shall examine the entities and relationships for the address book in prose.

The low-level model we shall be using is the relational model, which has been around since 1970 and is the model used by most commercial relational databases, including Oracle, Sybase and Informix.

Finally, we shall define the ways in which our users can see the data. Once we've completed this, we're done with the hard part and can move on to the tedious part—the implementation.

## Entities and Relationships

Since the entity-relationship model is such a high-level model, some of the hairier issues of the low-level model are not yet apparent, and our data model looks quite simple. The main advantage of the E-R model is that it presents a clear picture of the database miniworld—the segment of the real world that we're modeling—and is easily comprehensible to both laypersons and software engineers. Entities are defined as follows:

- **user**: This describes an address book user. This entity has a unique ID, a login name, a password, a "real name" and an e-mail address.
- **addressbook**: This is a "weak entity"--a set of persons for which a particular user has information.
- **person**: This describes a person for whom address information is available. This entity has a first name, a middle initial, a last name and a dynastic identifier (i.e., Jr., III, etc.). A person also has one or many of each of the following: e-mail address, postal address, telephone number and URL.

Relationships among entities are defined in this way:

- A **user** owns exactly one **addressbook**.
- An **address book** contains many **persons**.
- A **person** is an entry in exactly one **addressbook**.

If you've thought about databases before, you may be asking yourself, "Why can't a **person** be in more than one **addressbook**? Can't two **users** know the same **person**?" That is the sort of design decision you must make in this process —I chose to allow each user her own, private address book and avoid the difficult issues raised by sharing records. Allowing many people to share a record is like allowing global variables in a C program that can be modified by any function—you can easily get unexpected results when one function changes the variable's value without the others knowing about it.

## Moving to the Relational Model

Now we need to move our high-level model (which people can understand) to a low-level model (which our database management system can understand). This process is quite straightforward, although the database designer still has some options at this point, involving normalization. Normalization (see Resources) is a formal process to quantify and measure the quality of a relational model; as always, the tradeoff is theoretical quality versus performance.

The quick-and-dirty algorithm for moving your data model from the entity-relationship to the relational model is this:

1. Make sure every attribute of every entity is *atomic*.
2. Make a table for every attribute of an entity which can hold multiple values and define a join which ties these to the associated entity.
3. Make tables for each entity, using as fields the attributes you haven't dealt with yet. If an entity is involved in an n:1 relationship, include the key of the record it is related to as a foreign key.

This is necessarily a simplified version of the process; it does not deal with m:n relationships or some other details. For a more complete discussion of data model conversion, see a textbook on database theory.

SQL code for the “finished” relational model can be found at ftp.linuxjournal.com/pub/lj/listings/issue67/3475.tgz. All code is released under the GNU GPL.

### Views and Operations

The remaining design step is deciding what sort of capabilities we wish to grant users to access and update the data. This is perhaps not as much of a problem in our database, but what if we were designing a database of employees? It might cause great discord in the office if everyone knew the salary of the guy who spends every day surfing the Web and taking two-hour lunch breaks; however, they should be able to access his name, department and extension. Likewise, they shouldn't be able to change that information unless they are the department secretary or manager.

We do want *some* protection on our address book, so that you can type in your grandmother's e-mail address with the peace of mind that a spammer can't get it just by accessing your web server. We also don't want to bother the user with implementation details like unique ID numbers on each record—this should be a user-friendly address book. Therefore, we will allow the following:

- A user can retrieve records from her own address book.
- A user can insert and delete records in her own address book.
- The user will be shown only what she needs to see.

To this end, we create *views*. A view can be just a few columns of a table or a few columns of a join. In SQL, a view is defined with the **CREATE VIEW** statement, which creates a view from a **SELECT** statement. A view can be accessed just like a table, except you can't perform inserts, updates or deletes on it. Some of the views in our example application also use PostgreSQL

functions to make the final application programming easier, i.e., “make a mailto URL from this e-mail address”.

We also make note of the constraints which we cannot enforce with views: for example, the consideration that one may view only her own address book. We must implement these constraints in the application program.

### Implementation

Implementation in PHP3 is quite straightforward; many things in the example code speak for themselves, and others are well-commented.

The source code for the example application is intended to be more of a teaching tool than a finished product. It works well, but you would certainly want to add features before making a large-scale service from it. I have released it under the GNU GPL, so feel free to modify my code and share your modifications with others. This code is also on the FTP site shown above.

Resources

**Will Benton** can be reached at wcb@ccil.org

Archive Index Issue Table of Contents

Advanced search

# Automating IP Host Data Collection on a LAN

**Joe Nasal**

Issue #67, November 1999

Using Linux and the ACEDB database system makes easy work of local area network management.

Linux's agility and power inspires the efficient design and implementation of specialty tools for specific tasks. On a data network, engineers and administrators appreciate the ease and flexibility with which Linux can be implemented as a platform for data collection, analysis and processing. In this article I'll demonstrate techniques for implementing ACEDB (an object-oriented database) and a few other tools to provide comprehensive access to administrative data that you might already be collecting from your network.

The management of TCP/IP local area networks often entails an enduring struggle to control address space. Workstations, servers and managed subsystems (routers, firewalls, etc.) are all added to and subtracted from the network as the shape of the organization and the flow of data changes. However, remembering which machine is assigned to a particular IP address is not always as simple as keeping an up-to-date list of IP address to node assignments. Sometimes network architecture changes without proper documentation even under the best of circumstances—DHCP, BOOTP, ubiquitous SNMP and managed repeater ports included. Yet, keeping track of IP address assignments is important. Since the logical address space of an IP subnet is limited to a finite number of usable addresses, recycling IP addresses is a must. It's also useful to know the kind of machine responsible for a particular instance of packet generation when performing data-analysis on a LAN (metrics collection, troubleshooting, security audits, etc.). Automating the collection of and access to this information would go a long way toward reclaiming lost or unknown administrative data.

If you haven't yet discovered Arpwatch (available via anonymous ftp at ftp://ftp.ee.lbl.gov/arpwatch.tar.Z), please allow me to introduce you. From Arpwatch's man page, “Arpwatch keeps track of Ethernet and IP address

pairings. It syslogs activity and reports certain changes via e-mail." Arpwatch uses the libpcap API (ftp://ftp.ee.lbl.gov/libpcap.tar.Z) to listen for and capture ARP (address resolution protocol) requests and replies on a local Ethernet interface.

RFC 826 introduces the origins of the address resolution protocol, but you may find a more up-to-date description in your favorite networking handbook. For the purposes of this column, ARP is the method by which any machine on a logical TCP/IP subnet determines the Ethernet address (sometimes called a hardware or MAC (mandatory access control) address) of any other machine on the same logical subnet. ARP with respect to unicast data communications on a LAN provides a "sending node" with the Ethernet address of the "receiving node"--information essential to the successful completion of a communications transaction. The sending node generates an ARP request, a broadcast heard by all machines on the network asking for the machine assigned a specific IP address to respond with its Ethernet address. An ARP reply comes from the machine owning the particular IP address and contains its own Ethernet address as an answer to the query.

Arpwatch listens to the ARP conversations taking place between the machines on a network, extracts Ethernet address, IP address pairings from the dialogue, and stores the results with a timestamp in a local table. As ARP happens over time, this table grows to represent an accurate list of the available nodes on a network. Arpwatch cross references a machine's Ethernet address against a list of vendor descriptions of network interfaces, formats a report of its findings, and finally, e-mails the report or prints it to STDERR, like this:

```
From: arpwatch (Arpwatch)
To: root
Subject: new station (node.yourdomain.com)
   hostname:  node.yourdomain.com
   ip address:  192.168.10.10
   Ethernet address:  0:a0:24:56:c4:3a
   Ethernet vendor:  3com
   timestamp:  Sunday, May 9, 1999 11:16:57 -0400
```

Arpwatch only processes information collected from the same logical network that the interface it listens on participates in. This is true even if the LAN has been designed so that different subnets share the same physical wire for data transmission. Collecting data from multiple logical subnetworks requires the execution of separate instances of Arpwatch, one for each logical subnet, each tied to an autonomous network interface. Since Linux allows multiple network interfaces to reside in the same system (reference the "Mini-HOWTO on using multiple Ethernet adapters with Linux"), this can be conveniently accomplished in a single box. A modest PC with one Intel 486 DX-2 66 processor and an ISA bus can easily collect data from several busy subnetworks. Conversely, since Linux and Arpwatch are both readily obtained, multiple Arpwatch collection stations, one for each subnetwork, can be established if preferred.

Typically, Arpwatch is configured to distribute the information it collects via sendmail. This functions well as an alert mechanism with respect to changes within or additions to the network infrastructure, yet the utility of an Arpwatch report is therefore limited to the expression of this information in a static e-mail message. Sure, these reports can be searched and archived, but wouldn't it be useful to process and funnel the information collected by Arpwatch into a database? In this way Arpwatch's reports could be supplemented with a description of the node's responsible individual, its physical location, its operating system, its primary function or any number of other useful attributes.

The ACEDB database system was created by Richard Durbin and Jean Thierry-Mieg to provide a flexible and dynamic storage medium for a complex data set in support of their biological research. ACEDB is flexible in that it allows for a wide variety of different kinds of data to be stored. It is dynamic in that the structure of the database is easily modified as the data either comes to be understood differently over time or as it changes shape, such as through the addition of a new attribute or a new data type. ACEDB is also object-oriented, meaning that organization of data within the system preserves the real-world uniqueness and autonomy of each individual data object. Data within an ACEDB database is easily accessible because it is not sliced up into constituent parts and stored within relational tables. This provides for more intuitive access to data than is typical within many classic relational or object-relational database management systems. Don't get too hung-up on ACEDB's “object-oriented” tag. ACEDB's flair for objects simply means that it is easy to create and understand complex relationships between different kinds of data.

ACEDB is easily implemented within Linux. You'll find the latest distribution in the index at ftp://ftp.ncbi.nlm.nih.gov/repository/acedb/ace4/. Fetch a copy of the INSTALL script and examine the NOTES and README files for news and installation instructions. Download binary distributions of the ACEDB database and server, each of which has been precompiled for libc6.

Like Linux, ACEDB has enjoyed the benefits of collaborative engineering and development that are characteristic of an open-source software product. One of the major contributors to the ACEDB project is Lincoln Stein, perhaps best known for his work as the author of the CGI.pm module for Perl-based CGI scripting. Mr. Stein has created Jade, a Java-based interface between ACEDB and the Java programming language. He's also written an award-winning, object-oriented Perl interface for ACEDB called AcePerl. AcePerl provides functionality for connecting to and updating ACEDB databases, performing queries, fetching objects and other administrative tasks. To compliment AcePerl Mr. Stein has created and released AceBrowser, a small set of CGI scripts which work with AcePerl to provide for the easy creation of a web-browsable interface

to any ACEDB database. In this column, we'll use both AcePerl and AceBrowser to build, view and modify the database of IP host data collected by Arpwatch. You'll find links to the latest versions of AcePerl and AceBrowser at http://stein.cshl.org/. Download these distributions and make sure your local Perl is up to date (**perl -v** should report version 5.004_04 or higher). One goal of this project is to make the data stored in the ACEDB database available via the Web, so you'll also need a web server. Of course, the Apache web server is freely available from http://www.apache.org/. Finally, AceBrowser references CGI.pm, so make sure you've got version 2.46 or higher installed. CGI.pm is available at http://stein.cshl.org/WWW/software/CGI/.

We're going to use Linux to build a single system that functions as an Arpwatch collection station (a platform for storing the collected data in an ACEDB database), an ACEDB server and a web server for providing natural access to this data from anywhere on the network.

Install the ACEDB distribution via the INSTALL script. ACEDB comes with both a graphical (**xace**) and a text-based (**tace**) front end, each of which provide easy access to the system. Accessing an ACEDB database is as simple as running the **acedb** script from the top-level directory of an ACEDB installation and using the GUI to navigate through the data structure. Though ACEDB is powerful and highly customizable, a few other things must be known before getting started.

In an ACEDB installation, content of the files in the /wspec directory defines the ACEDB environment—everything from the appearance of xace's GUI interface and the allocation of cache space on the local disk to the internal definition of data structures. Rely on the default values presented in these files and verify that your user name is included in the passwd.wrm file in the list of authorized users. The key to understanding and getting started with ACEDB comes from an examination of the models.wrm, file also known as the “Model File”. The Model File is a template that defines both the structure of objects within the database and relationships that exists between objects. Each object in an ACEDB database has both a class and a name. An object's class describes the type of object that it is, and an object's name serves to uniquely identify it. Using ACEDB jargon, I could describe myself as an object of class “person” with the name of “Joe”. Or, if I were building a personnel database for a company, it might be more useful to describe myself as an object of class “Engineer” with a name of “NasalJS”, for example. I could make the object “NasalJS” more useful still by attaching a tag called “Full_name” to it with a value of “Joseph S. Nasal” and a tag called “Birthday” with a value of “06June1969”, so that people who use my database will know some additional information about “NasalJS”. ACEDB objects are built like this with a tree-like hierarchy of tags and values. The object file's content, called the schema, represents the generic structure (or form) of

each object class within an ACEDB database including the tags used to build objects and give them meaning.

The schema can be complex (as it is with the databases of genome sequences for which ACEDB was originally written) or simple depending on the data and the requirements of the application. We'll begin with a very simple schema to represent the data collected via Arpwatch. We can use the format of an Arpwatch report to suggest the structure of our ACEDB objects and create a schema that looks like this:

```
?Host   host_name UNIQUE Text
   ip_address UNIQUE Text
   ethernet_address current_ea UNIQUE Text
                        previous_ea Text
   ethernet_vendor  current_ev UNIQUE
      ?EtherType XREF participating_nodes
                        previous_ev Text
   timestamp current_ts UNIQUE Text
                  previous_ts Text
                  delta_ts Text
?EtherType   participating_nodes ?Host XREF
   ethernet_vendor
```

This schema defines two classes of objects, **Host** and **EtherType** (classes are identified using the syntax **?Classname**). In the **Host** class, the **host_name** tag has been constructed to contain a text value which is further described by the capitalized directive **UNIQUE**. Therefore, in an object of class **Host** the **host_name** tag will contain text and have no more than one data value. Similarly, the **ip_address** tag also utilizes the **UNIQUE** directive and may contain only data of type **Text**. Notice that the tags **ethernet_address**, **ethernet_vendor** and **timestamp** are each complex data types which are further described by their respective subtags.

An ACEDB database may utilize other data types (such as **Int**, **Float** and **DateType**, for example) but we'll keep things simple and treat all of the Arpwatch data as plain text. Directives other than **UNIQUE** are also available. In this example, the **XREF** directive is used to establish a cross-reference relationship between the **current_ev** subtag within the class **Host** and the **participating_nodes** tag within the class EtherType. The effect of this will be to build lists of **Host** objects which share a common network interface vendor.

Listing 1

To test this schema, we can fire-up xace via the acedb script and load a flat file containing sample data, as shown in Listing 1. In this example file, objects are separated by blank lines, and each tag of every object occupies its own row. Notice that when providing a value for complex data types, only the rightmost subtag needs to be specified. To test the schema, save this data in a file with an extension of .ace in the top-level directory of the ACEDB installation. Edit the

contents of the layout.wrm file in the /wspec directory to contain two lines, the first being **Host** and the second **EtherType**, so that xace will display these sample objects by default in the GUI. Start up xace and import the data file by selecting “Edit” and “Read an .ace file”. After opening the data file and reading in the data, use the GUI to navigate through the data set. Observe how objects are presented in accordance with the template established in the model file, including the results of the cross-reference relationship between objects of class **Host** and objects of class **EtherType**.

In the *live* system, we'll funnel Arpwatch records directly into the database using AcePerl. The first step is to capture Arpwatch's output to a file. Arpwatch's man page reads, “Starting Arpwatch with the **-d** flag inhibits forking into the background and e-mailing reports. Instead, they are sent to STDERR.” Using redirection, we can save Arpwatch's reports to a file, like this:

```
arpwatch -d > report.data
```

Listing 2

Perl is the perfect tool for extracting record data and building the database. In Listing 2, you'll find code which utilizes Mr. Stein's AcePerl modules to connect with an ACEDB server listening at port 20000100 on the local machine. After connecting with the server, the script drops into a loop of fetching Arpwatch records from the data file. When a complete record has been built, the last “else” condition in the while loop calls the “process” subroutine to update the database. First, the subroutine removes unnecessary whitespace and checks to see if **$host_name** contains the string **<unknown>** (if Arpwatch is unable to resolve the name associated with a node's IP address it marks the record **<unknown>**). Since we're going to use the data in **$host_name** to name ACEDB Host objects, the code translates **<unknown>** into a unique identifier based upon timestamp and the current Ethernet address.

Next, the data in **$host_name** and AcePerl's **fetch** method are used to see if a corresponding **Host** object exists. If it doesn't, a new **Host** object is created using AcePerl's **new** method, and the object is built by adding value to its tags with AcePerl's **add** method. Finally, the new **Host** object is written to the database with a call to AcePerl's **commit** method.

If it turns out that Arpwatch is only reporting some new information about an existing **Host** object, then **process** uses AcePerl's add and commit methods to simply update the existing object with the new data. Finally, the new record flag is reset and the subroutine exits back into the **while** loop to collect more record data.

When all of the records in the file have been objectified and added to the database, the script pauses for five minutes, then utilizes Perl's native **seek** function to reset the end-of-file error condition. This trick allows the code to follow and process the growing data file (emulating UNIX's **tail -f** command) as Arpwatch continues to collect more records.

As easy as that, we've built an ACEDB database by turning Arpwatch reports into objects and processing them with AcePerl. However, we've barely touched upon ACEDB's power for data representation and AcePerl's flexibility as an API for building and manipulating ACEDB objects. With a little bit more coding, for example, we could call methods within the SNMP Perl module to probe and collect data from SNMP-aware devices on the network and add this data to **Host** objects as they are built or updated. Or, we could add subclasses to the **Host** object based upon machine type (server, router, firewall, etc.). Any functional modifications which require changes to object definitions within the schema are easily handled by ACEDB. Go ahead and make the change and then let ACEDB automatically update the structure of existing objects within the database.

The last step is to make this database of Arpwatch records available via the Web. Mr. Stein's AceBrowser CGI scripts provide an easy solution. Unpack and install AceBrowser in a subdirectory of your web server's CGI directory. AceBrowser comes with a set of scripts for fetching, displaying and interacting with text and static GIF images stored in an ACEDB database. AceBrowser code is provided as an excellent starting point for creating custom web interfaces to any ACEDB database. However, AceBrowser's model-independent scripts can be used to display our data right out of the box, without modification. Follow the instructions for supplying the information in the site-specific global definitions (the location of the ACEDB server, the HTML stylesheet, etc.) and fire-up your web browser. I've used the **simple** search script to discover the **Host** object presented in Figure 1. The object is displayed using AceBrowser's **tree** script and is represented on-screen to mirror the object's structure as defined in the Model File. Notice that the data in the **current_ev** tag is represented as a hotlink. The cross-reference relationship defined in the schema creates lists of **Host** objects which share a common network interface vendor. In this example, the hotlink leads to a browsable list of hosts which use network interfaces manufactured by 3Com Corporation.

With ease and only a little bit of coding, we've used Linux and the ACEDB database to create a powerful tool for the collection of IP administrative data on a LAN. Linux and ACEDB are a good match because they are both flexible enough to allow for the invention of specialized databases on the fly and powerful enough to ensure data integrity. Lincoln Stein's AcePerl modules provide a powerful and friendly API for interacting with an ACEDB database,

and his AceBrowser scripts are ready-made to interface with any ACEDB database over the Web.

So what are you waiting for? What can you make Linux and ACEDB do?

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue67/3517.tgz.

**Joe Nasal** (joe.nasal@usa.net) lives to hack Linux and networks and often gets to do both as a software/systems engineer. However, he needs to escape from the lab more often to spend time with his beautiful wife, Kristin, who is eternally patient.

Archive Index Issue Table of Contents

Advanced search

# Buy One, Get One Free

**R. Scott Gray**

**Luke Pargiter**

**Phil Pfeiffer**

Issue #67, November 1999

Configuring Linux as a dual-purpose user environment.

In an era of decreasing budgets and increasing costs, it is sometimes impossible to follow standard wisdom and use distinct computer systems to achieve different goals. The East Tennessee State University (ETSU) Department of Computer and Information Sciences confronted this problem in spring 1997, when course requirements suggested the need for two computer laboratories:

- a stable computer laboratory that would support standard classroom use, and
- an experimental, Linux-based computer laboratory (LBCL) that would allow students to change operating system source code.

The obvious solution, which involved establishing separate laboratories, could not be implemented due to space and budgetary limitations. Therefore, ETSU system administrator Luke Pargiter decided to establish a single networked laboratory that would operate as an experimental or a stable platform.

Pargiter's strategy for supporting a dual-purpose laboratory partitioned the available computers into one trusted PC and a set of client PCs. The trusted PC, known as the *kernel server* PC, acted as a secure file server for the network as a whole. The client PCs were set aside for user development. Users were directed

to use only client PCs, and to boot these PCs according to how they intended to use the systems:

- Those who wanted to use the standard Linux kernel would use a designated bootstrap floppy that directed a user PC to download the standard Linux kernel from the kernel server.
- Those who wanted to use an experimental Linux kernel would bootstrap their PCs from the local hard drive, where they could store experimental kernels.

Pargiter's dual-boot strategy ensured the integrity of the standard network file system, while making it easier for an experimental user who created a bad kernel—and, consequently, rendered a PC unbootable from the local hard drive—to recover by rebooting from the kernel server PC.

### Configuration Difficulties

Configuring the six-station network posed two key challenges. The first one stemmed from the need to use surplus PCs with Micro Channel Architecture (MCA) buses—a type of bus the standard Linux 2.0 kernel does not support. The MCA problem was solved by using Chris Beauregard's MCA kernel patch from the Micro Channel Linux home page.

A second configuration-related challenge arose from the need to conserve disk space on the kernel server PC. Configuring the kernel server PC as a trusted server would help ensure the integrity of the kernel server's file system in the face of random kernel changes applied to satellite PCs. The server, however, would then be required to host one copy of the Linux operating system for every networked PC, leaving little room for user home directories.

The Linux kernel “disk bloat” problem was solved by observing that most of the files in the Linux distribution could be stored once on the server and shared among multiple PCs. This set of shared kernel files included system-independent files like /etc/passwd, which remain constant across the configuration, and satellite-independent files like /etc/exports that need to be configured only twice: once for the kernel server and once for all client PCs.

Since Linux distributions are not ordinarily partitioned into system-dependent and system-independent files, system-dependent configuration files were moved to a new directory, /adm. Most of the relocated files were originally stored in /etc. For each file moved to /adm, a corresponding soft link pointing to that file was created in each client's /etc directory. Trial and error was used to determine whether a configuration file could be moved to /adm.

## Server Mass Storage Device Configuration

The principal goal of the mass storage device configuration was to provide adequate disk space for system executables and files, while reserving the greater portion of disk space for user files. Initially, three mass storage devices were found on the kernel server: a CD-ROM, a 320MB hard drive and a 545MB hard drive.

The 320MB hard drive was configured as the default boot drive /dev/sda. This drive was partitioned into a 50MB swap partition, /dev/sda1, and a second partition, /dev/sda2, for the server's root file system. The first 80MB of the 545MB hard drive, /dev/sdb1, were reserved to house the client's operating system images, /tftpboot/Client_IP_Address, and the directory containing shared operating systems file, /tfpboot/adm. The remainder of the drive, /dev/sdb2, was set aside for the server's /usr/local directory. In retrospect, it would have been better to have partitioned /dev/sda into three logical disks, reserving the third partition for temporary files such as those found in the /var directory since the demand for temporary file space changes in response to a system's tasks.

The CD-ROM drive was included in the server's initial hardware configuration to simplify and speed up the Linux installation. One of the last configuration steps was to replace the CD-ROM with a 320MB hard drive. This third hard drive, /dev/sdc, was initialized with one partition, /dev/sdc1, intended for user home directories, /home. The CD-ROM was then placed on a client system, lin2, and exported to allow other clients and the server to utilize the device.

## Server Base System Installation

The MCA Slackware 3.1 installation boot floppy was used to initiate the server's installation. Basic Slackware 3.1 installation instructions were followed. The standard Linux 2.0 kernel series does not provide support for the MCA bus architecture; therefore, the kernel image residing on the MCA Slackware 3.1 installation floppy was installed on the new system. Upon completion of the Linux installation, the system was booted and the MCA Linux kernel patch applied to the kernel source code. The modified kernel source was then configured, compiled and installed. Finally, the server was rebooted to load the new kernel image.

## Network Preparation/Configuration

The basic network setup was configured during the Slackware 3.1 installation as with any standard Linux system. At the time, the Linux 2.0 kernel series did not support token-ring adapters. However, token-ring adapter support was provided via the MCA kernel patch. During the kernel configuration process, the

following networking support and network device support options were selected:

- TCP/IP networking
- drop source routed frames
- reverse ARP
- allow large windows
- network device support
- token-ring driver support
- IBM tropic chipset-based adapter support

The Slackware network initialization script, /etc/rc.d/rc.inet1, was edited to set up the token-ring interface (tr0). This was done by replacing the default network interface argument Ethernet, eth0, supplied to the **ifconfig** command with tr0.

A non-standard shell script, **rarptab**, was created and placed in the /etc directory. This script initialized the RARP (reverse address resolution protocol) table with mappings from 48-bit MAC (mandatory access control) addresses to IP addresses. The script compensated for the Slackware RARP's failure to respond to clients named in standard Linux initialization scripts. The rarptab script consists of a single command for each client of the form

```
rarp -a Client_IP_Address Client_MAC_Address
```

The rarptab script was invoked at system boot from /etc/rc.local, using the command

```
sh -c /etc/rarptab
```

The server's /etc/exports file was modified to give client PCs access to the following directories:

- /adm: shared files moved from /etc
- /home: user home directories
- /root: superuser's home directory
- /shlib: loadable libraries
- /usr: system tools and user directories
- /var/X11R6: XFree86 shared libraries
- /var/openwin: Sun Window system shared libraries

Client PCs were given read-write access to /home and /root directories. Access to all other directories was restricted to read-only. Entries added to /etc/exports had the following format:

```
Shared Directory  Client Hostname \
        (rw, no_root_squash)
Shared
        (ro, no_root_squash)
```

The **no_root_squash** option specified above is important for diskless operation. This option allows a client's SETUID programs to gain access to root-only accessible system files.

The **tftp** entry in the file /etc/inetd.conf was uncommented and the following line inserted in the file "/etc/services":

```
tftp 68/tcp #TFTP server
```

### Client Base System Installation

A directory, /tftpboot/Client_IP_Address, was created on the server to house the files for a single client system. A soft link, lin2, pointing to the Client_IP_Address directory, was created in /tftpboot for convenience. The /tftpboot/adm directory was then created to hold cross-system configuration files. Slackware's installation routine setup was invoked, and /tftpboot/lin2 specified when prompted for an installation target. Only packages from Section A (base Linux) and Section N (networking) were selected. During the configuration of the client's system files, redundant packages from Sections A and N were removed with ease. The setup routine option for installing the Linux kernel was refused, since kernels, by design, are loaded from specially prepared boot floppies. The setup option to run LILO was also refused.

### Client System Files Configuration

Listing 1.

The file /etc/fstab is used to determine which file systems are to be mounted at system boot. Remote clients on the LBLC share the common fstab file shown in Listing 1.

Sharable system files residing in /tftpboot/lin2/etc were removed, and soft links made to their counterparts in the /tftpboot/adm directory. The file /tftpboot/lin2/rc.inet1 was edited to ensure initialization of the token-ring interface (tr0). All file system integrity checks were then removed from /tftpboot/lin/2etc/rc.d/rc.S. The server must ensure file system integrity, since the **fsck** utility cannot be used by the client PCs to check file system integrity at system boot; at the time such a check would occur, the file systems have already been mounted.

Selected directories in which the contents were to reside only on the server (NFS mounted) were removed from /tftpboot/Client_IP_Address, using the following commands:

```
rm -rf /tftpboot/linwin2/var/X11R6/*
rm -rf /tftpboot/linwin2/user/*
rm -rf /tftpboot/linwin2/root/*
```

As discussed earlier, only a limited Linux installation was placed in /tftpboot/lin2; therefore, the following directories were created to serve as mount points to allow lin2 to import these directories from the server:

- /tftpboot/lin2/var/openwin
- /tftpboot/lin2/shlib
- /tftpboot/lin2/man
- /tftpboot/lin2/info

lin2's /etc/exports file was then edited to allow lin2 to export the CD-ROM that would later be placed on that machine. The following lines were added to lin2's /etc/exports:

```
/cdrom   lin1(rw,no_root_squash)
/cdrom     lin3(rw,no_root_squash)
/cdrom   lin4(rw,no_root_squash)
/cdrom   lin5(rw,no_root_squash)
```

The **chmod -R +r /tftpboot/lin2** command was used to make all files in the lin2 directory tree readable by **tftp**, a protocol that, essentially, can access only world-readable files. The configuration for lin2 was then used to configure lin3 through lin5. A duplicate of the lin2 configuration was created for each of the remaining clients with the commands:

```
mkdir /tftpboot/Client_IP_Address
ln -s /tftpboot/Client_IP_Address\
/tftpboot/"lin3 through lin5"
cp -Rpd /tftpboot/lin2\
/tftpboot/"lin3 through lin5"
```

A few of the system configuration files were then modified to make these files specific to their target hosts. The most notable changes involved modifications to /tftpboot/Client_IP_Address/etc/rc.d/rc.inet1. Specifically, two lines in the rc.inet1 file were modified to reflect the remote system's IP address and gateway:

```
IPADDR="Client_IP_Address" GATEWAY="Client_IP_Address"
```

Additionally, the file /tftpboot/Client_IP_Hostname/etc/HOSTNAME was modified to reflect each remote system's correct host name. Finally, the file /etc/exports was modified such that the /cdrom directory was no longer exported.

### Creation of Floppy Disks for Remote Bootstrapping

A two-step procedure was used to generate remote bootstrap floppy disks for client machines. First, the kernel source residing on the server was recompiled to include “root file system on NFS (Network File System)” support. Once

compilation was complete, the newly created kernel image was moved from /usr/src/linux/arch/i386/boot/zImage to /tftpboot/adm/client_boot_image_remote. A new disk was labeled and inserted in the server's floppy drive. A dummy device, /dev/boot22, was created with the **mknod** utility. The **dd** utility was used to copy **client_boot_image_remote** to the floppy disk. The **rdev** utility was then used to set the root device for the kernel residing on the floppy image to the dummy device /dev/boot22. The following sequence of commands was used to create the remote bootstrap floppy disk:

```
mknod /dev/boot255 c 0 255
dd if=/adm/client_boot_image_remote/zImage of=/dev/fd0
rdev /dev/fd0 /dev/boot255
```

Finally, this procedure was captured in a shell script and used to create additional remote bootstrap floppy disks.

### Local Client Configuration

All client systems included a hard disk configured to provide a single 50MB swap partition, /dev/sda1. This would allow experimental users to partition the remainder of the drive to suit their needs when installing Linux on their systems. Because the only CD-ROM resides on lin2, any installation of Linux to a client system other than lin2 has to be done via NFS. To make such an installation possible, modifications to the MCA Slackware 3.1 boot floppy were necessary. The boot floppy was mounted under the /mnt directory using the command

```
mount -t minix /dev/fd0 /mnt
```

The file /mnt/etc/networks was updated, changing LOCALNET to IP address 151.141.99.0. The MCA Slackware 3.1 boot floppy assumes that if a network installation is being conducted, the network is of type Ethernet. Therefore, it was necessary to edit the file /mnt/usr/lib/set/INSNFS, changing all occurrences of eth0 to tr0. As with the remote bootstrap floppy, a script was written to automate the above operations.

### Conclusion

The declining cost of disk drives, together with concerns about system security, have made diskless workstation operation and remote protocols like RARP less attractive. “Relatively inexpensive”, however, is still not quite “free”. We described a strategy for implementing a Linux-based laboratory that, in effect, uses diskless operation to support regular as well as standard users, at no additional cost.

ETSU's Linux-based computer lab has not been fully integrated into the ETSU curriculum, due primarily to unforeseen delays including equipment outages

caused by a thunderstorm. The lab, however, is operational. Two students used the lab in spring 1998 to develop a serial line driver for a senior-level operating systems course. Currently, plans are being made to use the laboratory as a tool for teaching graduate operating systems and undergraduate networking courses in the coming academic year.

**Scott Gray** (sgray@eolian.com) is the Senior System Designer for Eolian Inc. (www.eolian.com), where he is currently working on Internet caching proxy server development. Scott is also completing a master's thesis at East Tennessee State University on disk caching and adaptive block rearrangement. He and his cat, Sheba, regularly commute between their home in Church Hill, Tennessee and Eolian's office in Fairfax, Virginia.

**Luke Pargiter** is currently the Network Operations Supervisor at Pilot Corporation, a $2 billion privately owned petroleum retailer and travel center operator. He was formerly the Systems Manager for the Dept. of Computer and Information Sciences at East Tennessee State University. In his spare time, he reads, plays golf and guitar, and takes long walks with his wife, Rose. Luke can be reached via e-mail at pargitel@pilotcorp.com.

**Phil Pfeiffer**, assistant professor of computer and information science, currently teaches at East Tennessee State University (www.etsu-cs.edu). Phil, who earned his Ph.D. in computer science from UW-Madison in 1991, is currently working on his cave diving certifications. He has also done systems programming for PPG Industries (1977-1984) and taught computer science at East Stroudsburg University (1991-1996).

Archive Index Issue Table of Contents

Advanced search

# Linux in the Tropics

**Rodrigo Bernardo Pimentel**

Issue #67, November 1999

Linux is flourishing in Brazil. There is a Brazilian distribution, several high-traffic mailing lists, articles on mainstream newspapers and many users. Now, they have their first Linux User's Association.

As in the rest of the world, Linux is doing quite well in Brazil, thank you. The rapid increase in the number of users is stimulated by the availability of a Red Hat-based Brazilian distribution (http://www.conectiva.com.br/) with translated documentation, internationalized programs and an installation manual in Portuguese.

There are several Linux-related mailing lists, the largest of which (http://linux-br.conectiva.com.br/) has, as of May '99, more than 2000 subscribers. Linux is receiving extensive coverage in the Brazilian media, from technical publications to mainstream newspapers and business-oriented magazines.

Since Linux users tend to wander in herds, many Linux Users Groups (LUGs) can be found throughout the country. From time to time, there's discussion about a national LUG, a difficult task on which no concrete actions have been taken so far. Or hadn't until April 24th, when the first official São Paulo Linux User's Association (LinuxSP) meeting took place. We hope it was the first step toward a national association.

## What is LinuxSP?

LinuxSP is the first Linux users association in Brazil. Its home is São Paulo, Brazil's biggest city.

In Brazil, creating an association requires much paperwork and usually the payment of some fees. LUGs, therefore, are “informal” in the sense that they are not registered anywhere. With all the trouble it takes to register an association, why did we bother?

Certain advantages of an association appealed to us. To begin with, it sounds respectable. With Linux rapidly taking over the business world, companies want a name they can rely on. Reporters want a phone number to call (or an e-mail address to contact) when they need to write about Linux. In short, we intend to be a reference for Linux-related subjects in São Paulo and in Brazil. As an association, we can also legally receive donations and register domains (such as linuxsp.org.br), which can be done only by registered “entities” (associations, corporations, etc.).

Apart from that, LinuxSP is basically a LUG; we'll promote installfests, lectures, Linux courses, etc. Above all, we want to put Linux users in contact with each other. It is not our intention to “rule the Brazilian Linux universe”. We'll support the LUGs as much as we can with the infrastructure we've built—not replace them.

## A Bit of History

The original idea for LinuxSP started in late '97, with an unsuccessful meeting of Linux users in São Paulo. To put it simply, four people showed up. We formed a small LUG of our own and started meeting regularly. In '98, we decided to try again, this time with much more success (over 100 people came). Our next “enterprise”, an installfest, succeeded in bringing many people and failed in every other respect, except we learned a hard lesson about organization.

Finally, after another meeting, we gathered a group of people interested in leading the creation of a Linux users association. After much discussion on all aspects of having an association, it was finally legally created. As a landmark, we called a meeting to be LinuxSP's first official one. We estimated around 500 people had passed through during the day. When we counted our guest book, more than 850 had been present! That showed us LinuxSP had come just in time. As far as we know, it was the biggest Linux event in Brazil up to that time.

## How LinuxSP Works

The association is based on work groups. Every member of the association can join one or more groups, each of which focuses on a certain area, such as events, development, documentation, etc. Access to work groups is not limited to members; anyone who wishes can join the groups. LinuxSP was created for the Linux community, not just its members.

## What Now?

LinuxSP was very well received in São Paulo and elsewhere in Brazil. Many people signed up for membership at the first meeting, and e-mails just keep coming. If you're planning on starting something similar, have everything ready

for twice as many people as you expect to show up. We were surprised by the response to LinuxSP, and that caused a bit of delay in some changes we needed to make in order to accommodate all the people who wanted to join us.

The next step now is actually getting things done. The work groups are starting to get organized and we hope to have the association walking on its own feet (as opposed to having the 12 founding members organizing everything) very soon. The Brazilian Linux community has accepted us.

For more information, visit http://www.linuxsp.org.br/ in Portuguese; if there's demand, we will have the site translated.

**Rodrigo Bernardo Pimentel** is a Physics student at the University of São Paulo, Brazil and the president of LinuxSP. He can be reached at rbp@pobox.com.

Archive Index Issue Table of Contents

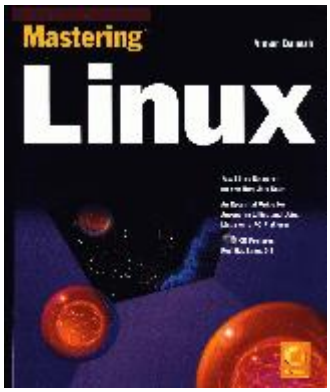
Advanced search

# Mastering Linux

**Bob van Poel**

Issue #67, November 1999

This book is, for the most part, well-organized and well-written, and contains much information; yet it falls short in its promise of mastery of the subject.




- Author: Arman Danesh
- Publisher: SYBEX Inc.
- E-mail: info@sybex.com
- URL: http://www.sybex.com/
- Price: $39.99 US (softbound, with CD)
- ISBN: 0-7821-2341-4
- Reviewer: Bob van der Poel

With Linux becoming more and more popular with home computer users and the media, as well as the traditional base of “hackers” and hobbyists, it is no surprise that a plethora of introductory books have appeared on your local bookstore's shelves. Some are excellent, some are poor, and some fall in between. *Mastering Linux* by Arman Danesh could be one of the good books, but due to some oversights, it doesn't quite make it.

*Mastering Linux* is a big book—928 pages; unfortunately, both its size and its title promise more than it delivers. It is, for the most part, well-organized and

well-written, and contains much information; yet it falls short in its promise of mastery of the subject.

According to the introduction, the book “aims to open the world of Linux to the average computer user.” It recommends that the reader be “comfortable using a Windows or Macintosh system” and be conversant with using the DOS prompt. The level of writing and the assumptions of prior computer knowledge in the book match these initial guidelines. The writing style is clear and easy to follow.

The book is organized into three major sections, with a large appendix.

The first section, “Welcome to Linux”, covers a bit of the history of Linux, an overview of the major distributions available, and the minimum hardware requirements needed for a usable system. Unfortunately, the author does not mention the large contribution made by the GNU project. I'm not suggesting we slavishly insist on calling Linux systems “GNU/Linux” or “Linux/GNU”, but fair is fair. No wonder Richard Stallman gets upset these days.

Section two, “Essential Linux”, guides the user though a typical installation from the supplied CD which contains a complete Red Hat 5.1 distribution. Next, the author covers the task of installing and configuring the X Window System and various window managers, printers, modems and some commands and applications. I believe the user described in the introduction should be able to get a working Linux system set up using these instructions.

Section three, “Linux in the Small Office/Home Office”, covers network configuration, Windows and Novell integration, setting up routers, web servers and sendmail. Most users needing to set up these types of services should have no problem using the author's instructions.

Finally, there is the appendix. At just under 300 pages, this is a major part of the book, but I'm not sure how useful much of it is. For example, Appendix B (18 pages) is a listing of all the fonts available under X (this seems to be a printout of “xlsfonts”). Appendix D (26 pages) is a listing of a sendmail.cf file. Also included in the appendices are “The Linux Hardware Compatibility HOWTO” and “The GNU Public License”. On a positive note, there is an excellent command reference which gives a short overview of nearly 200 of the more common commands supplied in a Linux distribution. In my opinion, this overview should have been combined into the main text instead of being hidden in an appendix.

My biggest problem with *Mastering Linux* is it appears to be incomplete in many subjects. Being a tad simplistic may be a virtue in a book destined for

“dummies”; however, something that promises you will “learn everything Linux has to offer your business or home office” has to rise well beyond the simple. In many places, the book either assumes too much of the reader, or it stops short of truly completing the task at hand.

Often, the book could leave the novice confused. For example, when introducing Linux commands and file names, no mention is made of the fact that Linux is case sensitive. To add confusion, the sections on the various commands are introduced with the command names in mixed-case, while the examples use the proper lowercase names. Even though some mixed case file names are used in the examples, presumably to illustrate that both upper and lower case letters are acceptable in a file name, no mention of the need to reproduce the case exactly is noted. Certainly, an experienced UNIX user will know this, but pity the poor tyro graduating from the DOS prompt.

At other places in the book, the instruction stops short of the “mastery level” promised. The section on setting up a PPP connection starts off using commands as root. This is fine for getting a PPP connection established for the first time. But, the author should have detailed methods for setting up the connection scripts so that users other than root can use them. On a properly configured system, there is no need to establish PPP connections only as the root user.

Another example of not going far enough is the section on configuring Sendmail. The author covers setting up sendmail just as an on-line server, since it is “a bit easier to configure and understand than off-line servers.” Yes, it is easier, but if I'm mastering Linux, I really would like to do learn some of the hard stuff as well.

The book does a good job in giving many overviews of the different major user packages available. As two examples, Chapter 6 has nice overviews of nine different window managers and Chapter 27 overviews ten different web servers. Similar comparisons are made throughout the book.

*Mastering Linux* is not a bad book. If anything, it is overly ambitious. It will enable a computer user with some basic computer skills to get up and running with a Linux installation on a PC, but I'm afraid some of its major oversights will also leave him frustrated or rushing to buy other books. It is useful, but don't be misled into thinking that this is the only book you'll ever need or that you'll master Linux with it.

Bob van der Poel started using computers in 1982. He has written and marketed many programs for the OS9 operating system. He lives with his wife, two cats and Tora the wonder dog on a small acreage in British Columbia, Canada. You can reach him via e-mail at bvdpoel@kootenay.com.

Archive Index Issue Table of Contents

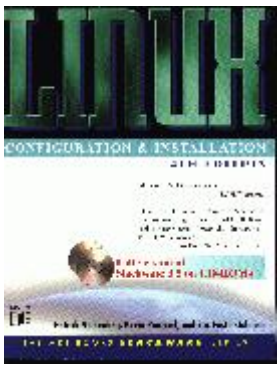
Advanced search

# Linux Configuration & Installation, 4th Edition

**Bob van Poel**

Issue #67, November 1999

Even though this book covers installation of a Slackware Linux distribution, I do think it contains much that will benefit new users as well as advanced users installing different distributions.




- Authors: Patrick Volkerding, Kevin Reichard and Eric Foster-Johnson
- Publisher: IDG
- E-mail: siteemail@idgbooks.com
- URL: http://www.idgbooks.com/
- Price: US $39.99 (softbound, with 2 CDs)
- ISBN: 0-7645-7005-6
- Reviewer: Bob van der Poel

It seems like only yesterday I sent off an order to Walnut Creek for the Slackware 96 CDs. When I received the four-CD set, it came with a tiny installation manual which I followed to the last detail. A few hours later, I had Linux (kernel 2.0.0) running on my PC. Since that eventful day, I have not reinstalled Linux on my main system, so I'm still running the same system. Thus, I have a bias toward Slackware, as it is what I use.

I have played with other distributions. Red Hat is no harder or easier to install, and SuSE comes with a much more novice-friendly manual. I'd recommend

Slackware to those who have more than a passing knowledge of computers and operating systems.

*Linux Configuration & Installation* is an enhanced version of the pamphlet which is shipped with the Walnut Creek distributions—a much enhanced version. The 560-page book comes with two CD-ROMs. The first is the standard Slackware installation with the 2.0.34 kernel—not the latest and greatest, but certainly a stable version. The second CD has a number of other packages, mostly with full source. Unlike previous versions of Slackware in which you need to create boot disks, you can now boot directly off the CD for your initial installation if your BIOS supports this, and most newer ones do. The CD can also be used as a rescue disk.

Chapter 1 (a full 43 pages) covers the hardware requirements for a successful installation. I'm sure if your hardware problem isn't covered here, you'll have great difficulty finding it anywhere else. I should also point out that this version of Linux is only for an Intel PC architecture.

Chapters 2 and 3 cover the actual installation of Linux on your PC and configuring the X Window System. Again, a good amount of detail is presented. In Chapter 4, the authors detail advanced issues: recompiling a custom kernel, printers and networking.

If you are using a laptop, you'll find Chapter 5, “Portable Computing Devices”, very useful. There is not a lot of text in this chapter; essentially, it just tells you that Linux works with most laptops. The useful part is a list of over 500 models and brands of laptops known to be compatible, and problems you can expect with some combinations. Many web resources are listed for optimizing specific models.

The remainder of the book, Chapters 6 to 9, covers Internet issues, a summary of basic Linux commands and applications and basic system administration.

My only complaint is some of the detail in the first half becomes almost painful to wade through. You may find it useful in certain situations, and as such, it is a good resource. I loaned the book and CDs to a friend who isn't a computer guru, but is mildly interested in trying out Linux. He's installed Windows 95/98 dozens of times, so has some knowledge. He spent a bit of time reading the book, decided it was all too complicated, and never even bothered trying to boot from the CD. I think my friend is more typical than many readers of this magazine might care to admit. I understand other distributions are addressing the fear of installation much better than Slackware.

The second part of the book, the various overviews, are an excellent introduction for people who haven't yet earned any UNIX merit badges. It will not make a wizard out of a new user, but should start them out on the correct path. I'm sure that if my friend had simply booted the disk, followed the on-screen instructions, then used this part of the book as a reference, he would have managed the install quite successfully. Quite correctly, the authors mention they are only touching on many topics and recommend further resources.

One actual weakness I discovered in the book was the way printers are handled. Setting up Ghostscript is covered in a short section, but there isn't any advice on handling some fairly basic and, to the new user, difficult problems. I can only assume the authors had an expensive PostScript printer hooked up to their own Linux boxes and were spared learning how to write “stair-step” filters and custom printcap files. Many printer problems are easily handled with print filters, but no mention of this is made.

Even though this book covers installation of a Slackware Linux distribution, I do think it contains much that will benefit new users as well as advanced users installing different distributions. I will certainly keep it as a reference. The authors certainly know their subject, but I have to wonder if all this detail is necessary.

Bob van der Poel (bvdpoel@kootenay.com) started using computers in 1982. He has written and marketed many programs for the OS9 operating system. He lives with his wife, two cats and Tora the wonder dog on a small acreage in British Columbia, Canada. Other interests are gardening, practicing sax and just having fun.

Archive Index Issue Table of Contents

Advanced search

# How to Install and Configure Oracle on Linux

**Greg Flannery**

Issue #67, November 1999

A step-by-step demonstration of the Oracle installation process.

Oracle RDBMS is a full-featured relational database management system from Oracle Corporation. It includes a set of administration tools, and precompilers for most programming languages. This article will cover how to install and configure an Oracle database on Linux.

The installation file for Oracle 8.0.5 Enterprise Edition is named Oracle8051EE_Intel.tgz. It can be downloaded from ftp://technet.oracle.com/. If needed, the glibc patch file glibcpatch.tgz can be downloaded from ftp://ftp.oracle.com/pub/www/otn/linux/.

Oracle currently uses glibc 2.0 rather than glibc 2.1. Oracle has supplied a patch which, when combined with compatibility RPM packages, allows it to run on Red Hat 6.0. I chose to use the Linux-Mandrake 6.0 distribution for this installation. If you are not using 6.0, you can skip the steps for installing the compatibility RPMs and the glibc patch. It is possible to install Oracle on other distributions, but since Oracle uses Red Hat for its development, Red Hat (or some variation of Red Hat) is preferred.

The C Development package must be installed when you install Linux. If you want to install Oracle's Intelligent agent, you must have tcl-8.0.3-20.i386.rpm installed. To install the JDBC drivers, you must define a path to your classesxxx.zip file.

I recommend creating a minimum of three partitions for Oracle. This allows you to use Oracle's Optimum Flexible Architecture (OFA) standard. Using OFA gives you the capability to segregate data from indexes and have multiple control files. The partitions should be named u01, u02 and u03. You can create more partitions if you have the disks to support them. I have two disks on my machine, so I created a partition of 1GB for /u01 and 150MB for /u02. I then

created a directory called /u03 to give me the equivalent of another mount point. The first partition (/u01) is where the Oracle executables and all associated files will be placed during the installation process. The remaining mount points will be used for data files, indexes and control files.

A multitude of directories are created during the installation process. Some of the more important ones are:

- $ORACLE_HOME/bin contains the executables for the database and administrative software.
- $ORACLE_HOME/rdbms/admin contains the SQL scripts used to create the catalog, and other useful scripts too numerous to cover here. Refer to the *Oracle Database Administration Manual* for an explanation of the scripts contained in this directory.
- $ORACLE_BASE/admin/SID/bdump contains the alert log for the instance. The name of the alert log is alert_SID.log, where SID is the system identifier for the instance (i.e., alert_greg.log for this installation). This file is extremely important in determining where problems lie. Any time you have an error or database crash, this is the first place to look for information on what might have happened to cause the problem.
- $ORACLE_HOME/network/admin contains the tnsnames.ora and listener.ora files. Both of these files are described in the section on modifying configuration files.
- $ORACLE_HOME/precomp/demo/proc contains several Pro*C example programs.

The data files, indexes and control files will be placed in /oradata/SID, where SID is the system identifier for the instance (i.e., /u01/oradata/greg, /u02/oradata/greg and /u03/oradata/greg for this installation).

The installation is broken down into 8 steps:

- Pre-installation sets up the groups and users, and performs all the tasks prior to actually installing the software.
- Software installation is the process of installing the software.
- Documentation installation is the process of installing the on-line documentation for Oracle. This must be done in a separate step due to a bug in the installation process.
- Binary patching is the process of applying the glibc patch supplied by Oracle.
- Database creation creates the initial database.
- Post-installation is the process of running a post-install script as root.

- Configuration file modification is the process of identifying and modifying the configuration files used by the database and Net8.
- Testing and automation is the final process of determining that the database is installed and configured properly, and setting the instance up to start automatically when the machine is rebooted.

### Pre-installation Steps

Download the compatibility RPMs from Red Hat. The necessary RPMs are compat-binutils-5_2-2_9_1_0_23_1_i386.rpm, compat-glibc-5_2-2_0_7_1_i386.rpm, compat-egcs-5_2-1_0_3a_1_i386.rpm, compat-egcs-c++-5_2-1_0_3a_1_i386.rpm and compat-libs-5_2-1_i386.rpm. Issue the following commands to install the RPMs:

```
rpm -ivh compat-binutils-5_2-2_9_1_0_23_1_i386.rpm
rpm -ivh compat-glibc-5_2-2_0_7_1_i386.rpm
rpm -ivh compat-egcs-5_2-1_0_3a_1_i386.rpm
rpm -ivh compat-egcs-c++-5_2-1_0_3a_1_i386.rpm
rpm -ivh compat-libs-5_2-1_i386.rpm
```

Edit /etc/passwd for root and change the shell from /bin/bash to /bin/sh. This will make the installation scripts supplied by Oracle run without errors.

Create the groups and users used by Oracle. At a minimum, you will need a group called **dba** for the oracle user. You may also want to create a group called **oper** for the operation of the database. Refer to the *Database Administrator's Guide for Oracle* to determine whether you want to create this group. In the following example, I used only the **dba** group, with a group ID of 601. You may need to use a different group ID if 601 is already in use on your system. Be sure to change the password for the **oracle** user.

```
groupadd -g 601 dba
useradd oracle -g 601
passwd oracle
```

Edit /etc/passwd as root to change the default shell.

Create the directories for your mount points if you didn't create them as partitions. Repeat the commands for all three mount points if necessary.

```
mkdir /u01
chown -R oracle.dba /u01
```

Change to the directory where you have unpacked the Oracle8051EE_Intel.tgz file (this is your staging directory for the installation process), and run the script to create the /etc/oratab file. This file is used by Oracle's startup script to determine which instances are running on the machine, and whether they should autostart when the machine is rebooted. More on this file later.

```
cd /home/oracle/orainst/orainst
ORACLE_OWNER=oracle; export ORACLE_OWNER
sh oratab.sh
```

See Figure 1 for prompts and their replies.

Log out as root and log in as **oracle**. Make sure **umask** is set to **022** by typing **umask**. If it is not set to **022**, you will need to add a line to your .profile file. Set the following environment variables in oracle's .profile:

```
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/8.0.5
ORACLE_SID=greg #(replace with your system
                #identifier)
ORACLE_TERM=386
PATH=$PATH:/u01/app/oracle/product/8.0.5/bin
TMPDIR=/var/tmp
export ORACLE_BASE ORACLE_HOME ORACLE_SID\
  ORACLE_TERM
export PATH TMPDIR
umask 022 #(only if the umask is not already
          #set to 022)
```

Log out, then log back in as **oracle**. Make sure the environment variables are set by using the **env** command, and that /bin:/usr/bin:/usr/local/bin is in your path.

### Software Install Process

Change to your staging directory and start the Oracle installer.

```
cd /home/oracle/orainst/orainst
./orainst /c
```

Select the custom install (see Figure 2). Read the preamble.txt file and press return to continue. Read the README.FIRST file and press return to continue. Select Install, Upgrade or De-install Software (see Figure 3). Select Install New Product—Do Not Create DB Objects (see Figure 4). Make sure the **ORACLE_BASE** and **ORACLE_HOME** environment variables are set correctly (see Figure 5). Press return to accept the defaults for your log files. It's a good idea to make a note of their locations, too. If anything goes wrong during the installation process, you may need to look at them. It is also a good idea to look at them even if everything works fine in order to learn more about the installation process (see Figure 6). Select Install from Staging Area (see Figure 7); make sure the staging area is set correctly. In my example, I used /home/oracle/orainst (see Figure 8). Select your language (see Figure 9). Press return to acknowledge the location of the root.sh post-install script. The default location is /u01/app/oracle/product/8.0.5/orainst/root.sh. Select the following products to install (see Figure 10):

- Client Software
- Net8

- TCP/IP Protocol Adapter
- Object Type Translator
- Oracle ConText Cartridge
- Oracle8 Objects Option
- Oracle8 Partitioning Option
- Oracle Unix Installer
- Oracle8 Enterprise (RDBMS)
- PL/SQL
- Pro*C/C++ (precompiler for C and C++)
- SQL*Plus

Use the space bar to select/deselect the products. Do not install the documentation. We will do that in a later step. After selecting all the products you want to install, tab to the Install prompt and press return. You will receive a warning about **ULIMIT** not being set. You can ignore this by pressing return (see Figure 11). Select **dba** as your DBA group (see Figure 12). If you decided to create a separate group for operation of the database, enter it for the **OSOPER** group (refer to snapshot Figure 13). For this install, I used the **dba** group as the **OSOPER** group.

Figure 14

At this point, the installer will begin copying the software. This is a good time to take a break, since this could take a while—how long will depend on the speed of your machine.

When the software installation process is complete, you should get a message that says “The requested action has been performed for selected products.” (See Figure 15) Press return to acknowledge the message, tab to Exit, then press return. tab to indicate Yes on the Confirmation and press return to exit the installer. The installer should exit with the message “Result: Success”.

### Documentation Installation

Make the directory where the on-line documentation will be installed (**mkdir /u01/app/oracle/doc**). Link the directory to correct a bug in the install process:

```
ln -s /u01/app/oracle/doc /u01/app/oracle/product/8.0.5
```

Start the installer from your staging directory (**./orainst /c**). Select custom install. Click on OK to bypass the README files. Select Install, Upgrade or De-install software. Select Add/Upgrade Software. Press enter to acknowledge the environment variable setting for ORACLE_HOME. Press enter to acknowledge the location of the log files. Select Install from Staging Area. Press enter to

acknowledge the path of the staging area. Select the appropriate language. Select Append to append to the root.sh script created during the software installation process. This is done because we haven't run the script yet, and we don't want to overlay the script created in the previous step. Press return at the Information screen, which gives the location for the root.sh script. Select the Intel LINUX Documentation and tab to install. Press return to begin the installation process. Press return to acknowledge the location of the on-line documentation (see Figure 16). Select the format (HTML or PDF) you want for the on-line documentation (see Figure 17). Once again, we're looking for the message, "The requested action has been performed for selected products." Press return to acknowledge the message, tab to Exit, then press return. Select Yes from the Confirmation screen to exit the installer. Again, we hope to see the installer exit with a message of "Result: Success".

### Binary Patching

Change to the directory where you downloaded the glibc patch file, glibcpatch.tgz. Unpack the file, **tar -xvzf glibcpatch.tgz**. Run the patch script glibcpatch.sh, **sh glibcpatch.sh**. The final screen should look something like Figure 18.

### Create the Database

Now it is time to create the original database.

- Change to your staging directory (**cd /home/oracle/orainst/orainst**).
- Start the installer (**./orainst /c**).
- Select Custom Install.
- Press enter twice to bypass the README files.
- Select Create/Upgrade Database objects.
- Select Create Database Objects.
- Press return to acknowledge the environment variables **ORACLE_HOME** and **ORACLE_BASE**.
- Press return to acknowledge the locations for the log files.
- Press return to acknowledge the environment variable **ORACLE_SID**.
- Select Oracle8 Enterprise (RDBMS), tab to INSTALL and press return.
- Select Create Product DB Objects (see Figure 19).
- Select Filesystem-based Database (see Figure 20).
- Select Yes to distribute control files over three mount points (see Figure 21).
- Enter the three mount points of /u01, /u02 and /u03 (see Figure 22).
- Select the appropriate character set.

- Select the appropriate national character set.
- Enter the password you want to use for the **SYSTEM** account. You will be asked to enter it a second time to confirm the password.
- Enter the password you want to use for the **SYS** account. You will be asked to confirm that too.
- If you want an internal password for **dba** and **operator**, tab to Yes at this prompt. If you don't want an internal password tab to No.
- Enter and confirm the password you want to use for the TNS listener.
- Click on No to configure the MTS Listener (see Figure 23).
- Press return to acknowledge the defaults for the location of the control files.
- Press return twice if you wish to accept the defaults for the paths to your data files and their sizes. If you have not done any database sizing and thereby determined you need more space, the default sizes should be appropriate (see Figure 24). You can add space to any data file at a later time, if necessary.
- Select Yes to accept the default file names and sizes (see Figure 25).

The installer will now create the initial database. As with the software installation, this is another good time to take a break. As before, we hope to see "The requested action has been performed for selected products." message. Press return to return to the main install screen; tab to Exit, then press return; select Yes at the confirmation screen.

### Post-Installation

Log out, then log back in again as root. Copy the oracle user's .profile to root's home directory (**cp ~oracle/.profile /root/.profile**). Log out, then back in as root. Check that the environment variables in the .profile are set properly by issuing an **env** command.

Change to the /orainst directory and run the root.sh script.

```
cd $ORACLE_HOME/orainst
sh root.sh
```

Verify **ORACLE_OWNER**, **ORACLE_HOME** and **ORACLE_SID** are correct. If they are, enter **Y**.

When it asks for the full path name to your local bin directory, enter /usr/local/bin. The script then tells you **ORACLE_HOME** does not match the home directory for **oracle**. This is not a problem. Type a **Y** and continue. The script will complete. (See Figure 26)

Log on as **oracle** and shut down the instance (see Figure 27).

```
svrmgrl
connect internal
shutdown
exit
```

### Configuration File Modification

Now for some cleanup and file modifications. The initSID.ora, where SID is the system identifier for the instance, file is located in the $ORACLE_HOME/dbs directory. This file is read by Oracle when the instance is started. It is used to set parameters for the instance, such as the amount of memory reserved for the database. There are too many parameters to go over in this article. Refer to the Oracle database administrator's guide for an explanation of the parameters and their recommended settings. You will probably be fine with the default values. However, if you have a large amount of memory on your machine, you may want to uncomment either the medium or large settings of the parameters in the initSID.ora file.

The oratab file is located in the /etc directory. This file is read by the dbstart file which we will use to automatically start the instance when the machine is rebooted. There are comments in the oratab file which explain the three fields and what they contain. Change the last field to Y for instances in which you want to start when the machine is rebooted. The file should look something like Listing 1.

Listing 1.

The listener.ora file is located in the $ORACLE_HOME/network/admin directory. This file is used by Net8 to determine how network connections are made to the instance(s) on your machine. Update the listener.ora file with the **sid** to which the Net8 listener should listen. Replace **oracle_sid** with the **sid** name. The file should look something like Listing 2.

Listing 2.

The tnsnames file is located in the $ORACLE_HOME/network/admin directory. This file is used by Net8 to determine the location for remote databases you can connect to. Replace **oracle_sid** with the sid name. The file should look something like Listing 3.

Listing 3.

As root, issue the following commands to set the permissions correctly for the Net8 files:

```
chown oracle.dba $ORACLE_HOME/bin/tnslsnr
chmod 750 $ORACLE_HOME/bin/tnslsnr
chown oracle.dba $ORACLE_HOME/network/log
chmod 775 $ORACLE_HOME/network/log
chown root.dba \
  $ORACLE_HOME/network/log/listener.log
chmod 664 $ORACLE_HOME/network/log/listener.log
```

If you receive an error because the listener.log doesn't exist, you will need to enter the last two commands after you stop and start the listener.

### Testing

Start the instance:

```
svrmgrl
connect internal
startup
exit
```

Connect to the database using SQL*Plus:

```
sqlplus
system
system_password
select count(*) from dba_objects; #(This should
 # return a count of the number of objects in the
 # database)
exit
```

Start the TNS listener:

```
lsnrctl start
```

You should see something like Figure 28.

Connect to the database using SQL*Plus through a network connection. This can be done using only one machine if you don't really have a network installed.

```
sqlplus system/system@greg
```

greg refers to the entry in the $ORACLE_HOME/network/admin/tnsnames.ora file

```
select count(*) from dba_objects;
exit
```

### Automation

Create the following symbolic links to automatically start and shut down the listener and Oracle instances:

```
ln -s/etc/rc.d/init.d/dbora /etc/rc.d/rc0.d/K10dbora
ln -s/etc/rc.d/init.d/dbora /etc/rc.d/rc2.d/S99dbora
ln -s/etc/rc.d/init.d/dbora /etc/rc.d/rc3.d/S99dbora
```

```
ln -s/etc/rc.d/init.d/dbora /etc/rc.d/rc5.d/S99dbora
ln -s/etc/rc.d/init.d/dbora /etc/rc.d/rc6.d/K10dbora
```

Listing 4.

Listing 5.

Listing 6.

Place the file dbora (Listing 4) in /etc/rc.d/init.d. Place the file lsnrstart (Listing 5) in the $ORACLE_HOME/bin directory. Place the file lsnrstop (Listing 6) in the $ORACLE_HOME/bin directory.

The listener and all Oracle instances designated to automatically start in the /etc/oratab file should shut down and restart when the machine is rebooted.

At this point, the database has been created. You can use SQL*Plus to create tables. If you are unfamiliar with SQL, there are a number of good books available on the subject.

### Oracle8i Install Differences

The install process for the latest release of Oracle for Linux (8.1.5 or Oracle8i) is slightly different. The installer has been rewritten in Java so the look and feel along with some of the responses are different. This section will cover the differences in the new install process.

You still need to create your dba group, oracle user, directories and mount points.

Download and install JRE (or JDK) 1.1.6 v5 from http://www.blackdown.org/. Create a symbolic link for the directory in which you installed JRE.

```
ln -s jre_install_location /usr/local/jre
```

Mount the CD-ROM which contains the Oracle8i software:

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

Log on as oracle and change directories to the CD-ROM and start the installer:

```
cd /mnt/cdrom
 ./runInstaller
```

You should see a welcome screen like Figure 29. Click “Next”. You will be prompted for the location of the installation jar file and your Oracle home directory. Make any necessary changes and click “Next” (Figure 30). Enter the dba group you created in the previous step (Figure 31) and click “Next”. You will

be prompted to run /tmp/OraInstall/orainstRoot.sh (Figure 32). After you run it, you should see the following lines of output:

```
Creating Oracle Inventory pointer file (/etc/oraInst.loc)
Changing group name of /u01/app/oracle/product/oraInventory to dba.
Return to the pop-up window and click Retry.
```

Figure 43

You will be prompted to install the Oracle8i Enterprise Server, Oracle8i Client or Oracle Programmer. Select the “Enterprise Serveri” (Figure 47) and click “Next”. You will be prompted for the type of install. Select “Custom” (Figure 33) and click “Next”. You will be prompted for which products you want to install (Figures 34, 35, 36). After you have selected the products to install, click “Next”. You can change the locations the products will be installed in or click “Next” to take the defaults (Figure 37). You will be prompted to create the database using the Oracle Database Configuration Assistant (DBCA). Select “Yes” and click “Next” (Figure 38). You will be prompted for the Global Database Name and the SID. Modify the screen capture to reflect your names (Figure 39). You will be prompted for the location of your database files. In my example, I used the mount point /u01 (Figure 40). You will prompted to select which network protocol(s) to install based on which protocols are present on your machine (Figure 41) click “Next”. You will see a summary of your install options. This will allow you to use the “Previous” button to change any settings that are incorrect (Figure 42). When you are ready to begin the install process click “Install”. The install screen will list where the log file from the install is being written (Figure 44). This information will come in handy if something goes wrong during the installation. When the install is complete, you will see a pop-up window (Figure 45). Note the location of the script to run as root, change to the directory where the root.sh script is located and run it. You may have to change the permissions on it to make it executable.

```
cd /u01/app/oracle/product/8.1.5
export ORACLE_OWNER=oracle
export ORACLE_SID=greg
chmod 700 root.sh
 ./root.sh
```

After the root.sh script successfully executes (expected output in Figure 46) return to the pop-up message and click “OK”. At this point the installation is complete, and you can click “Next”, then “Exit”. The testing and automation procedures are the same as in the previous section for Oracle 8.0.5.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue67/3572.tgz

**Greg Flannery** (gflanner@dha-us.com) is an Oracle DBA with nine years experience and three years experience with Linux. He lives in Atlanta, GA and works for Duley Hopkins and Associates as a Senior Member Technical Staff.

Archive Index Issue Table of Contents

Advanced search