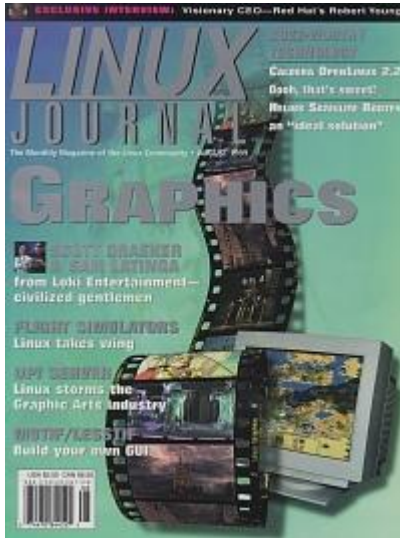


Advanced search

Linux Journal Issue #64/August 1999



Focus

Graphics by Marjorie Richardson

Features

Scott Draeker and Sam Latinga, Loki Entertainment by Michael J. Hammel

A talk with the company who brings you the computer game Civilization: Call To Power.

Motif/Lesstif Application Development by Glen Wiley

A tutorial designed to help you build your own GUI.

Linux as an OPI Server in the Graphic Arts Industry by Jeff Wall

A printing company finds Linux servers speed up their pre-press work.

Flight Simulators by Roman Melnyk

A look at Linux in the Aerospace Training Industry.

Forum

Linux Expo 1999 by Marjorie Richardson

Linux Journal attends Linux Expo.

Robert Young of Red Hat Software by Marjorie Richardson

Market Making for the Bazaar by Bernie Thompson

Hans L. Knobloch of IGEL by Marjorie Richardson

A talk with the head of the company that invented the thin client.

AbiWord: AbiSource's Open Source Word Processor by Craig Knudsen

A cross-platform commercial application is giving away their source—here's the story.

Dynamic Load Balancing DNS: dlbdns by *Harish V.C. and Brad Owens*

This article discusses an attempt to solve the problem of network traffic congestion by adding a dynamic load balancing feature to the existing DNS.

Columns

At the Forge Advanced “New” Labels by *Reuven M. Lerner*
Focus on Software by *David A. Bandel*

Linux Apprentice Graphical Toolkits for Linux Programs by *Patrick Lambert*

Graphical Toolkits for Linux Programs A brief look at several popular toolkits available for Linux.

Reviews

Caldera 2.2 Review by *Jason Kroll*

Helios Satellite Router by *Phil Hughes*

Programming Web Graphics with Perl and GNU Software by *Michael J. Hammel*

The Artists' Guide to the Gimp by *Syd Logan*

Departments

Letters

More Letters to the Editor

upFRONT

New Products

Best of Technical Support

Strictly On-line

Extending the Bash Prompt by *Giles Orr*

Terminal and xterm prompts can be created incorporating standard escape sequences to give user name, current working directory, time and more.

A High Availability Clustering Solution by *Phil Lewis*

Mr. Lewis tells us how he designed and implemented a simple high-availability solution for his company.

Introduction to Sybase, Part 3 by *Jay Sissom*

Plug and Play Hardware under Linux by *David Cantrell*

Open Source Remote Sensing Effort by *Dr. Shawana P. Johnson*

Remote sensing software is being developed using the Open Source model by the web project at remotesensing.org.

Linux: The Complete Reference, Second Edition by *Ben Crowder*

Archive Index

Advanced search

[Advanced search](#)

Focus: Graphics

Marjorie Richardson

Issue #64, August 1999

Everyone, including the fans of the command line, believe graphics are important to any operating system.

Graphics is always a fun issue. Everyone, including the fans of the command line, believe graphics are important to any operating system. With so many people learning to use computers on Windows or Macintosh systems, graphical interfaces are a necessity to make Linux applications accessible to these users. In past issues, we've had articles about KDE and GNOME, as well as developing GUIs with Java and CDE (common desktop environment). This month, we have a tutorial on building GUIs with Motif or Lesstif (Motif's freely available counterpoint).

Another reason graphics are important is games. Loki Entertainment has entered the gaming world by porting the popular Civilization game to Linux, and they have plans to port even more games. Michael Hammel talks to both the president and the lead programmer of Loki to find out what's happening with Linux in gaming.

We also explore the graphic arts industry and how Linux is being used here—a subject near and dear to our hearts. Finding that Linux is making inroads into pre-press departments is just the news we want to hear.

Elsewhere in this issue, we take a good look at Red Hat Software via an interview with Bob Young and a tour of their offices. We have also included a new section called "Up Front" to bring you bits of news about Linux and its proponents, quotes from Linux notables and kernels of information we think you will find interesting.

Marjorie Richardson, Editor in Chief

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

[Advanced search](#)

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

[Advanced search](#)

Interview: Scott Draeker and Sam Latinga, Loki

Entertainment

Michael J. Hammel

Issue #64, August 1999

A talk with the company who brings you the computer game Civilization: Call To Power.

The sound of marching soldiers thunders through the cobbled streets as citizens bask in the glory of their triumphant leaders. No, this isn't a post-election day Republican fantasy, it is just one of many scenes from the animated movies which introduce you to the first award-winning, top-selling computer game to be ported and run natively under Linux—Civilization: Call To Power.

Civilization: Call To Power is the latest in a series of games based on the original Civilization game. Civilization I hit the streets in 1991 and was an instant classic. In 1996 the sequel, Civilization II, won *PC Gamer's* Game of the Year award and was named the "Strategy Game of All Time". Together, Civilization I and II have sold over 2,500,000 copies, in an industry where a "hit" has to sell only 100,000 copies. Since its release in April, and up to the time of this writing, Civilization: Call To Power for Windows has been the number one selling PC game.

Figure 1.

This latest incarnation of the popular game was originally developed by Activision for MS Windows-based systems. The native X11 port to Linux was performed by Loki Entertainment Software, a relatively new company, started in 1998, devoted to porting the most popular PC games to Linux. As Scott Draeker (pictured at left below), President and Founder of Loki, says,

One way to put this success in perspective is to compare the PC game industry with the movie industry—both gross about \$5 billion a year. Getting

the rights to Civilization: Call To Power is like getting the rights to the next Star Wars sequel.

I talked with Scott and Sam Latinga (pictured at right below), lead programmer for Loki, as they prepared the Linux release of Civilization: Call To Power for distribution.

Michael: Although I'm a software developer by trade, I've never really been much of a game player, so this is somewhat new to me. Why don't you both give us some background information. How did you get into the world of gaming?

Scott: Well, I am not technical, but I am a game player. I came up through stints at Apple and LSI Logic and a few other places. After that, I went over to the "dark side"—law school. While working as a software licensing attorney at an accounting firm, I got involved with some game companies and was introduced to Linux. Dan Kegel at Activision introduced me to Sam.

Sam: My background is mostly in systems administration, but I've been interested in game system portability for a long time. I've worked on things like the Maelstrom port. When Scott came to me with the idea of porting games, I was a bit skeptical. When it became apparent to me that the idea was *for real*, my interest grew.

Figure 2.

Michael: What made you decide that now was the time to start porting games to Linux?

Scott: I first thought about games on Linux in 1998—much has happened since then. We were lucky and in the right place at the right time. When looking at Civilization, we were fairly sure we were going to lose money on it initially. Then, as I became more familiar with Linux, I heard about the GNOME and KDE projects. Knowing those sorts of familiar interfaces would be available along with the usual benefits of running on Linux convinced me a consumer market would develop for Linux and Linux products. I knew that if Red Hat could get a distribution on the shelf, I could certainly get a game there.

Figure 3.

Michael: The game was originally developed by Activision. Did you contact them about doing the port, or had they been looking for someone to do it? How much cajoling was involved in getting them to sign on?

Scott: I talked to many game companies, and I'm still doing so. We were actually focusing on another game when Dan Kegel suggested we look at Civilization. We did, looking at both demos and code. The more we looked at it and worked with Activision, the more we drifted away from the other game. I think Activision had the better title, and they certainly have their act together. Phillip Erwin, in the licensing department at Activision, was instrumental in getting the project completed. MicroProse, a Hasbro company and developer of the original Civilization, had to sign off on this port along with Activision. Phillip woke the President of Hasbro at midnight on December 31 in order to get the licensing signed. He's definitely gone to bat for us.

Figure 4.

Michael: Are you tied to Activision for future games, or are you free to talk with any other game producers?

Scott: We are already working on three additional titles, none of which is from Activision. Of course, I would love to do more Activision titles and we are pursuing that too.

Michael: Is this the same game as the PC version? Did the game play have to change at all in order for the Linux port to function?

Sam: Yes, it is the same game. The game play didn't change at all. However, we did add a few minor tweaks to the user interface which we feel make it a "real" Linux game. This is a true Linux application; that is, it's not a Windows game running under an emulator—it's native to Linux. The installation is native to Linux. (See "Game Review".)

Game Review

Michael: Were there any engineering limitations—libraries, for example, that you had to wait for before you could start the work?

Scott: Sam evaluated the Loki code, making sure their proprietary libraries could be either ported or emulated on Linux. The library we used is SDL, the Simple DirectMedia Layer. Sam authored SDL with an eye toward creating a cross-platform media layer for games. All the graphics and sound in the game have been ported to SDL, replacing the equivalent DirectX calls from the Windows version.

Figure 5.

Sam: The game also uses FreeType for font rendering. Activision did this on their own, which was really cool. It's one of the things they did to make the game more portable.

Figure 6.

Michael: So you had what you needed to get going?

Sam: Basically, we had all the tools we needed from the start, with a few exceptions. None of the available off-the-shelf installers and video playback tools were adequate for our needs.

We evaluated the available open-source video playback tools, brought them in, updated them and put them into the game with our modifications. We will release the playback tool back to the Open Source community.

Figure 7.

Michael: The community will appreciate that, I'm sure. How long did it take for this first port? Will it take as long for future ports, or did you spend some time during the original port building infrastructure for the porting efforts?

Scott: We started the port in December 1998, and coding began in January 1999. It didn't take that long—roughly four months. Part of the reason the port went so fast was the great support we received from Activision. They responded quickly to our questions and requests. With future games it may not take as long, but each game is unique. It depends on how tightly coupled the game is to the Windows API.

Gaming on Linux

Sam: Much of the user interface was provided in the game, so that part of porting wasn't difficult. However, in places where there was a tight tie-in to the Windows API, the port was a real pain.

Michael: I can imagine. When I got ready to install the pre-release version you sent me, I noticed that the game requires 400MB of disk space. Is it common to have such large installation requirements for games? Additionally, 43MB is a large footprint for a runtime binary from a Linux perspective. Will these sizes (installation and runtime) go down eventually?

Scott: Games are always hardware-intensive. The game binary is only 7.8MB. The rest is just as Activision designed it. Large installations (disk requirements) are common for games under Windows.

The game is designed to run from a CD, but due to space limitations on the CD, it can't currently be run that way. The files on the CD are compressed and would have to be uncompressed on the CD to run from there. We have no release plans for a CD-based Civilization, but we are aware of the need for such features and will be looking into that possibility for future games.

Michael: Any plans for hardware-accelerated graphic support? Have you talked to any of the X server vendors (XFree, Metro or Xi Graphics) about that?

Scott: Yes. We're talking to people on both sides—hardware and software—about hardware acceleration. We're working on a 3-D game right now that will be hardware-accelerated.

Civilization is truly a breakout product. It is a triple-A product released hot on the heels of the Windows version going into stores. That includes SuSE in Europe and Macmillan Publishing in North America. We intend to place it in all the big stores—all the places that sell software. Wherever you see Red Hat, you'll see our game. That's not true for the average Linux game at this time. By doing this, we've shown there is a market for this game, and games in general, on Linux. Many announcements are being made by 3Dfx, Creative and other people. They are coming on board because they realize games draw hardware sales.

Civilization Playing Tips

Michael: So there will be many changes in the gaming market down the road. With all those games coming, how hard is it to do these ports? What are the biggest areas of work in porting a game from Windows to Linux? Is it on the technical side or in the business details (including things like documentation)?

Sam: From the programmer's perspective, it's probably 50-50.

Scott: The programmers had a few things they were doing for the very first time: compiler debugging and a few infrastructure issues. Having done that work, we can leverage it in the future. On the business side, the first time is much harder. This is the first shrink-wrapped game for Linux that can be found in Best Buy, Fry's and CompUSA. A fair bit of work was needed just to contact game companies with both a great product in good shape for porting and an interest in exploring this market! In addition, we are working on three other products right now and that will just keep increasing. Much of our past work will carry over—it was necessary only because it was the first time.

Sam: Although each game will be unique, the business side stays more stable. It does not change much after the first game.

Scott: With Linux becoming more commercial, you have to justify your products to the free software community and to the business world. In the business world, you pay to put your product on the shelves, do co-marketing, and pay for ads, bundling and so forth. That's a great deal of work and an important part of legitimizing the Linux platform for other games.

Michael: Scott, do you think there is room in the Linux market for more than one game-porting company at this time?

Scott: I get the occasional e-mail from someone else who wants to port games to Linux. We know we're not going to be the only ones doing this. Several other games, not necessarily ports, are available already, for example, Hopkins FBI and Quake 3. We've picked our niche for now by doing top-20 games. Since so many games come out each year, it's self-evident that we cannot port everything or even everything considered good.

We came in fairly early to prove to the computer industry, the game market and the Linux market we could do this. We wanted to position ourselves as the people who are doing the triple-A products. It's not quick and dirty. It's not contract programming. We're taking the source, re-implementing it for Linux and publishing it for the Linux community.

We're a godsend for game companies. At a time when they're laying people off, we offer another market for games already making money for them.

Michael: Well, the game market for Linux is fairly large. But how do you find out where that market is and what they want? You mentioned in an interview with LinuxToday that you read *Slashdot* and *LinuxToday*. Do you feel these on-line Linux news sources are representative of the whole market for Linux games, or is there a larger market beyond the generally technical readers of sites like these?

Scott: There is a market right now just with current Linux users. As it grows, it will bring in more and more people, more and more diversity. Right now you can go to *Slashdot* and hit a good percentage of Linux users. In two or three years, that might not be the case—not because *Slashdot* has gotten smaller, but because the market is going to keep broadening. Right now, we focus on the technically savvy Linux market—there is a tar file on the CD, for example. On the other hand, we're aware that non-traditional users will be coming over. The GUI installer is there, specifically for the newbies.

Sam: I did the installer and am rather proud of it, since I wrote it specifically for the newbies.

Michael: I must admit it was quite easy to use. What about getting the games in the hands of users? I noticed GameCellar.com (<http://www.gamecellar.com/>) has a Linux area, and CTP is the only product there currently. The typical Linux user wouldn't look for a product here, however. Does this mean you see a different market than typical Linux users? Do you plan on selling through traditional channels (Linux Mall, LinuxBerg.com, through Red Hat or SuSE, etc.)?

Scott: A good deal of interest in Linux products is showing up in many places it hasn't been in the past. As an experiment, Costco made Red Hat available in their stores, and it outsold Windows 98. I understand you can even buy Linux products at some Wal-Mart's. These aren't traditional UNIX/Linux outlets. We're excited about where our games might show up. Obviously, it will be at the superstores and the game stores like Software Etc., but it will be in bookstores, too. A great synergy exists with bookstores, since they already sell Linux products. Selling our game will be an incredible add-on.

Michael: What's next? Do you have any concrete plans for future games?

Scott: No announced titles, but we have 3-D and strategy games in the works, and we're very close to doing an RPG (role playing game). We're going to hit as many genres as we can, with all the best titles. You won't see twenty RPGs, but you will see the best ones.

Michael: Seems like a good bit of work for a start-up. Do you have enough staff to work these projects in parallel?

Scott: We have enough to work two in parallel now, and we want to ramp up to three or four by the end of the year. We plan on having four to eight titles by the end of the year and doubling that next year.

Michael: Well, that's great. Thanks for taking the time to chat with me, Scott and Sam.

After we'd finished our telephone interview, Scott, Sam and I started talking candidly about playing the game. I mentioned to them how engrossed I'd become in the game during the week after I'd received the pre-release copy sent to me to prepare for this interview. I spent several nights glued to the computer until 3AM. Perhaps it's a good thing I live alone—where does one find a spouse who actually understands and accepts such behaviour? Sam chuckled at this. "We have an in-house joke at Loki that with games on Linux, we're going to bring kernel development to a standstill."

We all laughed a bit more about this possibility. I don't actually think kernel development will slow because of such high-powered games as Civilization: Call

To Power. Still, kernel developers may find they are getting fewer hours of sleep each night. I wonder—does this mean the next thing to come to Linux will be an open-source caffeine supplement?



Michael J. Hammel (mjhammel@graphics-muse.org) is a graphic artist wannabe, a writer and a software developer. He wanders the planet aimlessly in search of adventure, quiet beaches and an escape from the computers that dominate his life. Michael writes the monthly “Graphics Muse” column in Linux Gazette, maintains the Graphics Muse web site and the Linux Graphics Mini-HOWTO. His book, *The Artists' Guide to the GIMP*, is published by SSC, Inc.

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

[Advanced search](#)

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

Motif/Lesstif Application Development

Glen Wiley

Issue #64, August 1999

A tutorial designed to help you build your own GUI.

This article is directed toward those who have spent their time in the pleasant arena of development and are finding it harder and harder to ignore the growth of graphical front-ends. My perspective is that of a developer of commercial UNIX applications distributed largely on commercial versions of UNIX. I have an interest in making these applications easily portable to Linux and developing skills that are useful across OS boundaries. Given a choice, I prefer to use technologies which can be readily applied in both the commercial and open-source worlds.

My goal is to give you a basic understanding of the essential components of an X/Motif application—enough to begin developing your own applications. Development of server-side programs or command-line interfaces (CLIs) tends to be more cut-and-dried than that of graphical user interfaces (GUIs). Human beings are notoriously unpredictable, especially when one is used to developing according to strict protocols and well-defined boundaries. The complicated interaction with a GUI is usually best handled by a toolkit or library—someone else can worry about the details of where the mouse pointer happened to be when the left button was pressed, and how to render those beautiful 3-D effects on the components on the screen. Although every UNIX ships with libraries to take care of these details (X, Xt), they are still quite tedious and time-consuming. Motif is a library that provides enough insulation from the underlying primitives to make development of GUIs realistic, while still allowing easy access to those primitives should you need them.

Strictly speaking, Motif is a standard put forward by the Open Software Foundation (OSF) that describes expected behavior and the look and feel of an application. An application can be described as Motif-compliant; OSF even provides for branding of applications that adhere to the Motif standard. The practical utility of the standard is arguable; however, the benefits of the

framework provided by the Motif libraries is unquestionable. You can build a Motif application that uses the libraries but is not compliant with the standard; in fact, most programmers simply use the library and ignore the more Byzantine dictums of the standards.

Henceforth, when I say Motif, I mean the Motif library or any one of the various API-compatible libraries (Lesstif, Mootif, Moteeth, etc.). Differences between Motif (v. 1.2) and Lesstif will be pointed out. Although Motif 2.1 has been released, I will focus on version 1.2 since this is the revision level of most installed clients and the most common open-source alternative, Lesstif. The newer releases of Motif are generally backward compatible, although some techniques and elements of the API may be deprecated. The biggest differences between versions 1.2 and 2.1 include thread-safe libraries and some additional widgets, such as the Notebook.

Why Motif?

My decision to use Motif is motivated by several very specific concerns, some of which will strike a chord in those of you involved with commercial software development. The tools I use in commercial efforts must:

- be tried and proven—the less risk the better
- be readily available to the target end users
- not place undue burdens on development time
- have a reasonable learning curve (for my sake)
- have a reasonable licensing policy, preferably no additional run-time fee for the end users (for marketing's sake)
- utilize a development skill set that is relatively easy to find (have mercy on your manager)

The Motif library is a toolkit that meets my requirements. Virtually every commercial UNIX ships with a run-time license for Motif. Free UNIX variants (including Linux and BSD) also have access to Motif libraries (although at a fee); however, a Motif “clone” called Lesstif is now stable and mature enough to use in place of a “real” Motif library. Alternative GUI toolkits exist (GTK, Qt, etc.) and may be a good choice for projects which dodge the commercial world—for those of us who make a living writing software for commercial distributions, I think Motif is still a good choice.

The Common Desktop Environment (CDE) that ships with most commercial UNIX operating systems is built on Motif 2.1, which was intended to provide a backward-compatible API for applications built on version 1.2. I have found I can work on the same project on Solaris (using Motif) and Linux (using Lesstif), moving the source back and forth regularly with surprisingly few hiccups.

Finally, the end product looks sharp. Motif makes it easy to get a professional-looking GUI without any extra effort. 3-D effects, text clipboard operations (cut, copy and paste) and other trappings users have come to expect from even small applications are taken care of automatically, so you can focus on your application's problem domain.

Getting and Installing Motif/Lesstif

A Motif development license can be had from Red Hat software (<http://www.redhat.com/>) or the Open Group (<http://www.opengroup.org/desktop/motif/>). If you have to pay for the development license, I strongly recommend you use Lesstif and find a commercial workstation with a Motif run-time license already installed. Odds are, you can tweak any errant code without needing to purchase Motif. If you have a spare machine on which to install Solaris for the Intel platform, Sun now makes Solaris available for non-commercial use for the cost of the media (about \$10 US); this includes a run-time license for Motif and the headers needed for development. It is fairly easy to do your development on Linux/Lesstif, then make a quick build under Solaris/Motif to check compatibility.

Remember, though, Lesstif is an implementation of Motif version 1.2 with limited support for Motif 2.0, while the commercial implementation of Motif 2.1 has already been released. The good news is the vast majority of systems are still using Motif 1.2.

Lesstif can be downloaded from a number of places, beginning with <http://www.hungryprogrammers.org/>. Binaries are available for Linux (and some other UNIX flavors) as RPMs or gzipped tar files from the site and its mirrors. Installing Lesstif on my Red Hat-based workstation was as simple as typing:

```
rpm -i lesstif.rpm
```

You don't need the development RPM unless you plan on contributing to the Lesstif project.

For most commercial UNIX systems, the developer will not need to do anything extra to start developing using Motif; for example, Sun ships the Motif headers and libraries with Solaris. If you choose a commercial Motif license for Linux, you will need to follow the steps outlined in the package. Getting things rolling on Slackware or some other non-RPM-based system is rather trivial. The libXm files should end up in the `/usr/X11R6/lib` directory (or `/usr/dt/lib` on Solaris), and the headers should be placed in their own directory (named `/Xm`) off the `/X11` include directory, e.g., `/usr/X11R6/include/Xm` or `/usr/dt/include/Xm`.

X Components

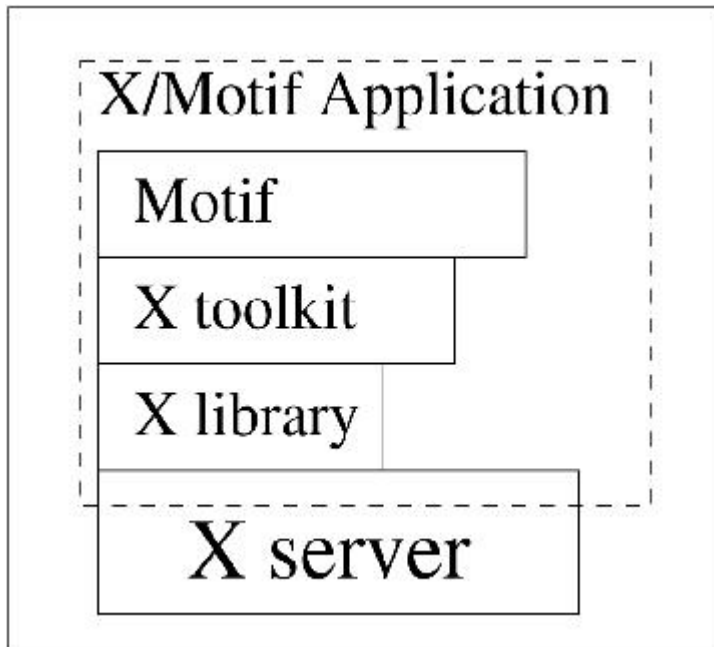


Figure 1. Accessing X with Motif

Access to the X Window System in a Motif application is accomplished primarily through three libraries: the X library (commonly referred to as Xlib) which provides the most primitive interface, the X Toolkit (Xt) upon which Motif is built and Motif (see Figure 1). A book covering Xt, or at least the *Motif Programming Manual*, would be a good investment if you become serious about GUI development. Many libraries choose to build on Xt rather than X, so it is reasonable to expect X and Xt to be consistent across UNIX flavors and available on any end-user configuration.

The X library functions are typically prefixed with “X”, and the X toolkit widgets and functions are prefixed with “Xt”. Although the toolkit is used fairly heavily within a Motif application, the need to access X library routines is rare. Symbols provided by Motif (functions, constants and variables) are prefixed with “Xm”.

The X server via the window manager is responsible for interacting with the video hardware and dispatching messages and events to the applications. It is important to note applications running under X are implemented using an event-driven paradigm. The programs generally spend their time waiting to receive messages that indicate such events as mouse movement, key presses, or exposed windows—more on this later.

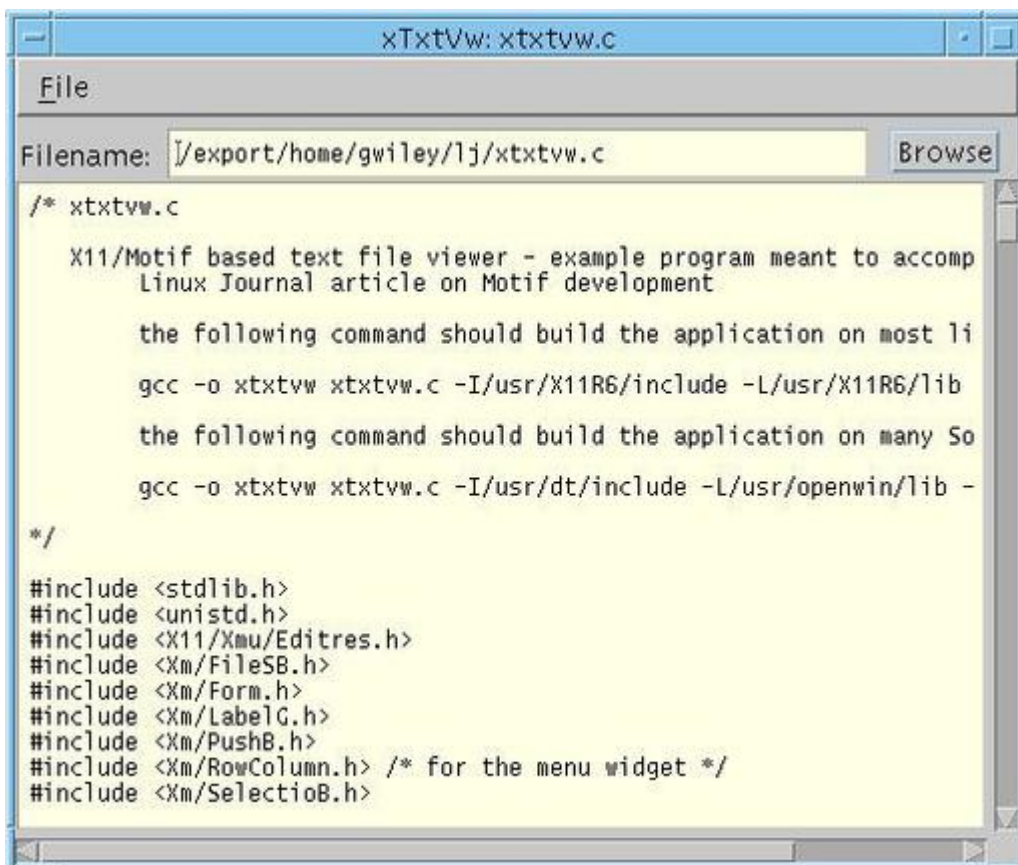
Graphic Primitives

The graphic components used to build an X/Motif application are called widgets. Some simple widgets are a push button, a radio button and a text field; more-involved widgets might be a file selection dialog or an HTML widget.

Many graphic components are composites of widgets coupled within a routine or C++ class. You could spend your career developing GUIs without ever learning how to create a custom widget. It is best to avoid writing custom widgets until you are very comfortable with the X/Motif environment; even then, you should have good reasons for not using more conventional techniques. The Motif widget libraries have been implemented in a very object-oriented fashion (especially for something written entirely in C). If this sort of thing interests you, I strongly encourage a study of the widget libraries (refer to the Lesstif source). Many widgets are available that implement functionality not addressed by the Motif library.

A number of generic routines can be used to create and alter instances of widgets (e.g., XtCreateManagedWidget); in addition, widgets typically make some specific routines available (e.g., XmCreateScrolledText). The run-time behavior of a widget is controlled through resources; that is, variables specific to a single instance of the widget which determine aspects of its behavior such as position, color, button labels, etc.

Getting Started



```
xTtxtVw: xttxtvw.c
File
Filename: /export/home/gwiley/lj/xttxtvw.c Browse
/* xttxtvw.c
   X11/Motif based text file viewer - example program meant to accomp
   Linux Journal article on Motif development

   the following command should build the application on most li
   gcc -o xttxtvw xttxtvw.c -I/usr/X11R6/include -L/usr/X11R6/lib
   the following command should build the application on many So
   gcc -o xttxtvw xttxtvw.c -I/usr/dt/include -L/usr/openwin/lib -
*/
#include <stdlib.h>
#include <unistd.h>
#include <X11/Xmu/Editres.h>
#include <Xm/FileSB.h>
#include <Xm/Form.h>
#include <Xm/LabelG.h>
#include <Xm/PushButton.h>
#include <Xm/RowColumn.h> /* for the menu widget */
#include <Xm/SelectioB.h>
```

Figure 2. Simple Text Viewer

A good way to get a grip on a new technology is by looking at something built with it. Motif provides a rich and extensible environment, and it would take far too long to explain all the details behind the steps I will take; in fact, I will just

scratch the surface of what can be done using Motif. The application I will use to illustrate Motif is a simple text-file viewer (see Figure 2). You should download the source (it's long) from ftp.linuxjournal.com/pub/lj/listings/issue64/3392.tgz. I will refer to specific line numbers as I explain different aspects of the Motif API.

Motif (and X) development takes advantage of “Event-Driven Programming” using an event-dispatch loop. An application must set up the interface (or at least some minimal components) before turning control over to the loop that handles events (via XtAppMainLoop). Once the event loop is invoked, your code will be called only in response to an event. The important thing to remember is that as long as your code is running, your application is not communicating with the X server. You must take care to return control to the event loop regularly, especially during time-intensive operations. Callbacks, which are functions intended to be invoked by the event loop in response to a specific message, are used heavily. Motif and the Xt library (upon which Motif is built) take care of the vast majority of the events an application might receive—the only ones you must handle are those for which the default behavior is inadequate.

Compiling a Motif application is not too involved—you must ensure the Motif header files are available and the libraries are listed in the correct order on the link command. The following command will build the sample application on Linux using Lesstif or Motif:

```
gcc -o txtvw txtvw.c -I/usr/X11R6/include\  
-L/usr/X11R6/lib -lXm -lXmu -lXt -lX11
```

To build the application on Solaris, issue the following command:

```
gcc -o txtvw txtvw.c -I/usr/dt/include\  
-L/usr/openwin/lib -lXm -lXmu -lXt -lX11
```

If these don't work, look for a directory called /include/Xm in the directory tree containing your X Window System and specify that directory. The libraries are typically in the X library directory—search for a file named libXm*. Motif is most often made available as a shared library; the version will vary depending on your system. If you cannot find one or the other (especially the headers), you might not have loaded the X or Motif development package.

Setting Up the Application

Let's start by examining the code used to initialize the application, in order to cover some of the essentials. A fair amount of work has been done to make life easier on developers of international software; localization and internationalization are addressed by a number of features available in the X

and Motif libraries. I will avoid treating this issue at all, except to say that you should make a habit of including a call to `XtSetLanguageProc` in all of your X applications.

The second toolkit function I call is `XtAppInitialize` (line 85), which has a few interesting calling arguments. This function is responsible for initializing the Xt library, evaluating any command-line arguments meant for the toolkit (such as geometry) and initializing X resources used by your application. **context** is a return parameter used by the X toolkit; you will not typically do much with this argument, apart from passing it to the main event loop. It is a work area for the libraries—a place for them to store state information about instances of widgets in your application. The second parameter is the **class** of your application. It is used to refer to resources that belong to the application by both internal and external clients, such as **editres**.

Listing 1.

The fifth and sixth arguments to `XtAppInitialize` are the argument count and argument list passed on the command line. The X toolkit will consume any command-line options intended for it, for example the location and initial size of the application's main window and the default background color. See the X11 man page to get a better idea of exactly which options are accepted. After this call, the remaining options can be processed by your application.

The other parameter of interest is the fallback resource string array. This is optional, but a good idea, as it specifies default values for resources used by the widgets in the application. Command-line options and any hard-coded values will override the settings listed here, but you should make a habit of specifying at least the background and foreground colors (and arguably the fonts) to help keep the interface appearance consistent.

The widget that gets created by `XtAppInitialize` is the application shell, containing the window that appears on the desktop as the application. This is a special widget which provides interaction with the window manager. Applications can have multiple shells if more than one primary window is required. This sample application has a single shell that ultimately parents all other widgets.

Application and Widget Resources

The application and the widgets used within that application are presented on the display, based on their resource values. Even a simple text label has a set of resources used by X to determine its appearance. It is probably best to think of resources as data members for a specific instance of a widget. Some examples of resources are colors, the text of a label or text-entry widget, size, location,

font and callback routines for events. Widgets are constructed in a hierarchy: each widget has a parent widget, and each widget may have zero or more children. Each resource is a name,value pair. The name is comprised of the application class, the name of each widget in the hierarchy and the actual resource name, in that order.

X and Motif define a number of primitive types which can be used as values for resources such as XmString (the Motif display string), Widget (any widget), Position (location) and XtPointer (used for passing data of any type to callback functions). Resource values can be retrieved and set using XtSetValue and XtGetValue. (There are also the XtVa* versions of these functions.) If you specify the wrong type for a value, you won't find out until runtime, so it is a good idea to verify the types using a man page or the reference manual. It is also important to keep an eye on your application's STDERR output for messages from the X libraries.

An example of a name for a Widget resource would be ".xtxtvw.title" (the text in the application window title bar). It could be specified in the ~/.Xdefaults file as the application resource "XTxtVw.xtxtvw.title". I encourage you to explore resource names using editres (see the section on debugging). The X server maintains a database of resource names for all applications. These specifications come from a number of sources, including the initialization file for your window manager (such as ~/.fvwmrc), the X defaults (~/.Xdefaults), application-specific defaults (on Linux /usr/X11R6/lib/X11/app-defaults, on Solaris /usr/openwin/lib/app-defaults) and runtime specification by the application.

The **xrdb** command provides a means for viewing and changing the resource settings in the database at runtime (changes affect only widgets created afterward), although editres is far easier to use for applications which support it. Take a look at the manual page for xrdb, but note it typically *replaces* the database settings. If you want to merge additional settings, you must specify this using the **-merge** option.

I specify some application resources in the fallback resources passed to the initialize function. A simple example of setting some widget resources can be seen in the setMainTitle routine.

Widgets

Some widgets are designed to handle layout and event management for other widgets. These are called manager widgets and include XmForm, XmBulletinBoard and XmRowColumn. A number of widgets derived from the manager widgets provide enhanced behavior, such as the XmSimpleMenuBar.

XmBulletinBoard provides the means for placement of widgets at specific locations—you must specify the X and Y offsets within the bulletin board. This widget is generally not used alone, but instead to derive other more useful widgets such as XmForm. XmForm allows you to specify the placement of widgets relative to others within a form. This is probably the most useful and commonly used manager, since it automatically handles resizing and dynamic changes in its children widgets at run-time.

When a widget is created, it is not actually displayed until it and every widget in the hierarchy above it is managed. Widgets created as children of an XmForm widget will not appear on the screen until the form widget is managed via a call to XtManageChild. This will cause every managed widget in the form to suddenly appear. Typically, you will want to create a manager widget as the single child of the top-level shell, then use that widget as a parent for the remaining GUI components (line 103).

Another widget used in the application is XmLabel (line 157). This is used for simple output of text strings and images that typically do not interact with the user. Thus, labels are often created as Gadgets, a special sort of widget that defers much of its event handling to its parent. This means fewer callbacks are registered, fewer events get dispatched and generally helps improve performance. I try to use gadgets when I do not expect a widget to interact with the user or when there are many instances of a particular widget (such as rows in a table). Push buttons (line 173) are derived from XmLabel.

The XmTextField widget (line 164) is used to provide a single-line data entry field or a more flexible output mechanism. The text field is a specialized sort of text widget intended for situations where the more fully featured XmText widget would be overkill. The XmText widget can do everything the XmTextField can, and more. The XmScrolledText widget (line 202) is used to display large amounts of text and is actually packed with enough features to be used as a nice text editor. In fact, you could implement a text editor using the example application by adding fewer than 20 lines of code.

Strings

Motif uses a special representation for strings that are intended to be used in the display—usually of type **XmString**. Many widget resources require values of type **XmString**, and the most common method for creating them is via a call to XmStringCreateLocalized (line 110). Note that storage is allocated for created strings, and it must be returned via a call to XmStringFree when no longer needed (line 124). If the application must be internationalized at a later time, strings constructed this way will be presented in the local language and typeface.

Strings can also be retrieved from widget resources using `XmStringGetLtoR` (line 277); this call allocates storage for a character array using memory management routines available in the Xt library (`XtMalloc`). Storage for strings created using this technique must be freed using `XtFree` (line 284). Some convenience functions provided by Motif, such as `XmTextGetString` (line 323), also use Xt memory management, and their results must be released using `XtFree`.

“Normal” character arrays are still used in a number of places in Motif, but it is important to recognize that `XmString` will pop up all over the place. This is another area where the loosely typed nature of C is going to rear its ugly head. Be careful about releasing storage associated with these functions—memory leaks in applications using these functions are quite common.

Creating Widgets

The menu bar created in line 114 illustrates some important issues; the most obvious is the variable argument list accepted by the function. The Xt and Motif functions rely heavily on variable argument lists; the advantage of this is simplified semantics, the disadvantage is no compile-time-type checking. A good defense against errors is to carefully columnize the argument lists in your source—it helps provide a visual check that the call is properly constructed. Some errors in the argument lists will show up on `STDERR` at runtime, and some will just pass quietly.

The first argument to the function is the widget that will parent the one being created; the second is the name of the widget as it will appear in the resource database (try to find this using `editres`). The remaining arguments are resource specifications as pairs of resources and their associated values. The entire list is always terminated by **NULL**.

An alternative method of creating widgets, used beginning with line 194, requires you to generate an argument list. The **create** call is similar to the variable-argument list in that the parent and widget name are the first two arguments, followed by the number of arguments in the list and the list itself. It is common practice to define an array of type **Arg** (this is from Xt) and reuse it in each call. Although the fixed-length argument list is a bit more cumbersome, some widgets do not provide a variable-argument create call. In addition, I have had trouble with **gcc** 2.7.2 and some of the variable-argument functions—it has something to do with macros used by Lesstif. If you receive odd complaints from the compiler on one of the variable-argument list functions, you may want to try converting it to the fixed-length argument list function. I have had none of these sorts of problems with the newer releases of gcc and Lesstif.

The resources not specified in the call to the create function will be valued from the resource database maintained by the X server and from defaults specific to the widget. You can also change them later by calling `XtSetValues` (line 376) or `XtVaSetValues` (line 181). In fact, this is the most common method employed to change aspects of the interface at runtime. Some Motif widgets also provide convenience functions to accomplish the same thing (such as `XmTextSetString`). They have the distinct advantage of providing compile-time-type checking and usually present simpler semantics.

Widgets are often created as unmanaged—this defers their display until they are managed. In fact, widgets are not managed by default and require a call to `XtManageChild` to get them to actually display. There is a way of building widgets which are managed from the time they are created, that is, by using the `XtVaCreateManagedWidget` and `XtCreateManagedWidget` functions (line 157). Since this is a call to the Xt library, the order of the initial arguments differs slightly from the Motif syntax. The first is the name of the widget (normally the second argument in the Motif create functions); the second argument is the widget class, a constant that matches the name in the Motif create function with the first character in lower case. The third argument is the parent widget, which is followed by the resource names and values.

Menu Details

Motif provides a simple menu bar that is easy to build but makes some assumptions about menu behavior. For a more flexible widget with more involved menus, you will want to explore `XmMenuBar`. Note that the menu bar is created as a child of the form. The menu bar built on line 114 specifies a single menu as a cascade button (line 115). If you want multiple pull-down menus, you simply add more lines. The string is the name of the menu as it will appear in the menu bar, and the single letter is the meta keystroke that will cause the menu to pull down (in this case, **alt-F** or **meta-F**).

Once the menu bar is constructed, the actual pull-down menus can be created as children of the menu-bar widget. Line 133 specifies the parent widget and name for the “File” menu; the arguments indicate the initial item and the callback function invoked if the user selects a menu item. Each of the subsequent lines describes a single menu item and each item begins with its type. The Menu can contain more than just the traditional push buttons; for more information, refer to the *Motif Programming Manual*. The next argument is the string displayed for the menu item, followed by a hot key the user can press when the menu is displayed, then an accelerator key sequence the user can press to invoke the menu item without actually pulling down the menu. The last argument for the item is the display string for the accelerator keystroke—this will be displayed right-justified in the menu.

Horizontal lines (separators) can be placed in the menu by specifying XmSEPARATOR for the type (with no associated values) as in line 135. This is another case where columnizing your source is a *very* good idea. It is easy to leave out one of the items, resulting in catastrophic (or at least bizarre) behavior at runtime. If you don't want to specify one of the options, a place holder such as an empty string must be used.

Once all the pull-down menus are created, the menu bar must be managed (line 142), so that when the form gets managed, it will be free to display the menu bar. In this case (as with some of the specialty manager widgets), it is not necessary to manage the pull-down menus. Remember, they aren't actually displayed until the user does something to make them appear.

File Selection Dialog

The next widget we instantiate is the file selection dialog. It will not actually appear until invoked by the user (by a menu selection or button press), but I chose to build it here since it is logically related to the file menu. Motif provides a number of very nice specialty dialog boxes; one of the most interesting is the File Selection Dialog, XmFileSelectionDialog. This widget is derived from XmSelectionBox, so when you are looking for some of the resources in the *Motif Reference Manual*, you may need to refer to both widgets.

At times, it may be more convenient to assign callbacks after creating a widget, either because you don't have all the information you need at the time, or the callbacks cannot be easily supplied to the create function as in our example. Another reason is the explicit call to XtAddCallback makes it obvious the assignment took place; in addition, you get the benefit of compile-time-type checking.

The stock file-selection dialog handles all heavy lifting for you, file name globing, directory navigation and even file name filtering. The default dialog also provides a few extra buttons I will not use. To avoid confusing the user, set their XmNSensitive resource to false (causing them to be grayed out) or remove them altogether. Motif provides a convenience function for all widgets derived from the Selection Box to get the widget ID of specific children widgets in the dialog (line 150), XmSelectionBoxGetChild. Once I have the widget ID, a call to XtUnmanageChild will prevent that particular widget from appearing when the rest of the dialog is displayed.

Form Attachments

The code on lines 157 to 204 is concerned with adding the remaining components to the display. I will use these widgets to explain the way XmForm handles the layout of its children. Attachments describe a widget's relationship

with other widgets within the form; it may take a while to get used to the way attachments work. Widgets appear in the form in the order they are created—important to keep in mind as you code. I will describe the most basic aspects of form attachments.

Each widget parented by a form has attachment resources for the top, bottom, left and right sides. I will cover how to use `XmATTACH_FORM` and `XmATTACH_WIDGET` and attachment offsets. Other attachment types include `XmATTACH_NONE`, `XmATTACH_SELF`, `XmATTACH_POSITION` and some based on form and widget opposing sides. On line 116, you see the top, left and right sides of the menu bar are attached to the form. This means each of the sides will “stick” to the respective sides of the form, stretching the widget as needed. The bottom attachment is left as `XmATTACH_NONE`, since I need to use the remaining portion of the form for other display elements. If the attachments were not specified, the size and position of the menu bar would default to a fairly unattractive box in the upper left corner of the form.

On line 158, I specify that the top of the label is attached to the menu bar widget and line 159 specifies this attachment will stay 10 pixels off the menu. The left side of the label is attached to the form and the right and bottom sides are left unattached. If attachments were specified and a border was visible for the label, you would see it stretch as the position of the other widgets changed during a resize operation.

Mistakes in attachments can hide a widget altogether or generate ugly screens; in some cases, the errors are bad enough to be printed on `STDERR` (e.g., circular dependencies). If you specify a widget attachment, you must also specify the widget to which the side should be attached. Since no compile-time checking is done on these things, you won't discover your omission until runtime.

I want the text field used to enter a file name to stretch horizontally; so once the Browse button is created, I add a final attachment to the text field (line 181). This is an excellent example of the strength of using `XmForm` for your layouts.

Exposing the Interface

Once the interface has been constructed, a call to `XtRealizeWidget` (line 215) for the application's top-level shell will cause all of the managed widgets in the hierarchy parented by that shell to display. Widgets can be constructed and destroyed at runtime. You are not required to create them all before exposing them; in fact, in some cases you will not have enough information to do so, but as much as possible of the main screen of the application should appear on startup.

The call to `XtAppMainLoop` (line 217) relinquishes control to the event dispatch loop. The only way for code in your application to execute from this point is in response to an event. Many events (such as `expose` and `hide`) are handled automatically by the Xt library. It is generally a good idea to assume `XtAppMainLoop` will never return. Applications commonly provide a routine to handle shutdown tasks such as closing files and cleaning up temporary workspaces. This routine would be called from the “Exit” menu item, for example.

The callback functions installed for the various widgets will actually give life to your application. Mundane tasks such as reacting to mouse button presses and keyboard input are all handled by the libraries supporting the instantiated widgets. This can make the application challenging to debug, since the flow of control is rather flexible, and it underscores the importance of careful design. Callbacks that call other functions invoking actions of other widgets—you get the idea. There is no easy way to automatically generate a call graph with this sort of implementation.

If your application needs to do something time consuming, it may be best to use a co-process, or at the very least, issue regular calls to `XmUpdateDisplay` (as on line 293). This will prevent the GUI from appearing locked. A number of different approaches for implementing progress bars and other user feedback are also available. You must remember that your application can react to the user's input only when it is not executing your code.

Callback Functions

Callbacks can be intimidating to deal with initially, but they are nothing more than pointers to functions placed in an event table. When the event for which they are registered occurs, the event dispatch loop invokes them with a few parameters. Since callbacks are passed limited state information, much of that information must be retrieved from the environment.

Callbacks are associated with events explicitly or implicitly. The `XmVaCreateSimplePulldownMenu` call on line 133 implicitly registers `fileCB` to be called whenever the user selects a menu item. The call to `XtAddCallback` on line 147 registers `fileselCB` to be called when the OK button in the file selection dialog box is pressed.

When Xt or Motif invokes a callback, it passes the widget for which the callback was registered, a single piece of client-specified data and a callback data structure specific to the widget and event for which the callback was invoked. When a callback is added to a widget via `XtAddCallback`, the final argument to the function is user-specified data of type **XtPointer**. This gives you a hook to pass information via pointers to structures. The user data element would then

be cast back to the original type within the callback—be very careful here, unless you enjoy generating core files.

The **call_data** parameter is usually an instance of a callback structure derived from a basic callback structure—the first few elements in all of the `Xm*CallbackStruct` types are the same. For `XmFileSelectionBox`, the structure is defined as:

```
typedef struct
{
    int      reason;
    XEvent   * event;
    XmString value;
    int      length;
    XmString mask;
    int      mask_length;
    XmString dir;
    int      dir_length;
    XmString pattern;
    int      pattern_length;
} XmFileSelectionBoxCallbackStruct;
```

The meaning of each element is explained in detail in the *Motif Reference Manual*, and I make use of the **reason** and **value** elements on lines 275 and 277. The **reason** tells you why the callback was invoked; this is especially important with something like the file selection box, since it is common to use the same callback for multiple events. The **value** member is the actual file that was selected (its complete path). A quick look at an instance of a file selection box on the screen will give you a good idea of how the other elements are used. If you are not sure about the type of callback structure at runtime, you should cast it to type **XmAnyCallbackStruct**. This at least has the **reason** and **event** members which can be used to determine the nature of the call.

Debugging and Experimenting

Debugging Motif applications can be challenging, due to the heavy use of callback functions. I find the trusty **printf** is often the quickest way to ferret out anomalies. The situation is not improved by the way the library relies on type casting to implement the object-oriented design of the widget libraries. Refer to documentation on function signatures unless you are absolutely certain you have it right.

A program called `editres` provides a mechanism to quickly tweak resource settings at runtime. By pointing the program to your running application, you can retrieve the widget tree and examine or change resources for nearly every widget instance. This is a great way to experiment with alternative fonts, colors, attachments and labels. Applications that are properly linked can speak the `editres` protocol, hosting a two-way communications link to the outside world without changing a single line of code.

In order to add support for editres to your application, add the following two lines:

```
#include <X11/Xmu/Editres.h>
XtAddEventHandler(g_topshell, (EventMask) 0, \
True, \
(XtEventHandler) _XEditResCheckMessages, 0);
```

and link against the Xmu library—place it after the Motif library on the link command. Kenton Lee has a good article on using editres at www.rahul.net/kenton/editres.html. The source for editres is included with X distributions, so any machine running X should have the utility.

Alternative Approaches

Coding an application using the technique described here can be a bit tedious at times, although it yields the most flexibility and control over the interface. There are many different approaches to developing Motif applications, including wrapper libraries and GUI builders. Two of the most common are the User Interface Language (UIL) and the Motif Tools library. Both reduce the amount of energy that goes into coding, and each has different advantages.

UIL is a C-like description language used to create an interface implemented via Motif and other Xt widgets. The UIL files are compiled and fed to the Motif resource manager (MRM) at runtime; the MRM then renders the interface, installs callbacks, etc. The chief advantages of UIL are rapid layout of screens, extensive error checking, easier internationalization and less complex code. The main disadvantages of UIL are reduced flexibility (due to the language's simplicity), which could require additional coding in Motif, yet another language to learn, and instability problems. Another issue for the aspiring Lesstif developer is that UIL/MRM is not yet fully supported.

Motif Tools (Xmt) is a library that came with the O'Reilly book, *Motif Tools* by David Flanagan. This library provides a few additional widgets and a significant number of support functions, letting you program using resource files. This approach has the advantage of providing more runtime flexibility, since the resource files are read at runtime. Minor changes to the GUI may not require you to recompile, and end users can edit the resource files to alter the interface to their tastes. In addition, application code can be significantly simpler, since you rely on Xmt to do the detailed implementation work. Xmt does have a fairly steep learning curve and may still lack some of the flexibility provided by coding directly with the Motif API.

I stuck to C in this article, since there was so much I wanted to cover; however, C++ is also a good fit for Motif development. The biggest issue is that callback functions must be static member functions or global functions. In fact, user

interfaces are particularly well-suited for GUI development, since you have visual objects that can be represented by the C++ objects. Also, the encapsulation C++ brings to the table can make the programs more modular and easier to maintain.

Resources



Glen Wiley is a Senior Software Engineer for 3Com Corporation's Carrier Systems—Network Management Systems Research and Development. He began programming around 1985 and was first introduced to UNIX in 1987 via an AT&T 3B2 running System V. When not trying to cram more stuff into his overtaxed brain, he spends time as a human trampoline for his two sons Stephen and Richard. He can be reached at gwiley@ieee.org.

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

[Advanced search](#)

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

Linux as an OPI Server for the Graphic Arts Industry

Jeff Wall

Issue #64, August 1999

A printing company finds Linux servers speed up their pre-press work.

Recent testing of high-performance file-sharing tools for the Graphic Arts industry has shown Linux to be the top performer. We tested Linux against the Macintosh OS X Server, the Sun Solaris and the Windows NT server. Linux came out on top, yielding a blazing average of 8800KB/sec throughput over 100Base-T Ethernet. We have been using an NT server in this role for two years, but it has been stumbling under the load placed on it in the last year. Now, we restart the server each day in order to avoid hanging the print services.

As a full-service commercial printing company, Mahaffey's Quality Printing is one of more than 30 percent of the 44,000 U.S. printing firms which use or could use an OPI (open pre-press interface) server to generate and spool high-resolution PostScript output files to RIPs (Raster Image Processors). The OPI server, a vertical market application, could be described as an industrial-strength file and print server specific to the graphic arts industry. It costs from \$5,000 to \$25,000 or more to obtain one for use in a very large printing plant with up to 100 pre-press client workstations.

Using an OPI server allows a client station to offload the processor and I/O-intensive work of creating a high-resolution PostScript file. The OPI server creates and then forwards the PostScript file to the RIP. RIPs are powerful workstations that convert huge PostScript files into ultra-high-resolution laser recorder instruction sets. These files are imaged to film or plate and ultimately used on offset printing presses.

The need for an OPI server begins at the creative client machine. This is usually a Macintosh, since most content creation in the printing and publishing industry is done on Macintosh workstations. Most modern printing or publishing pre-press departments are multi-platform client/server-based, with a UNIX or Windows NT departmental workgroup server that serves Macintosh

and Windows client workstations image files and pages as needed. It also consists of numerous PostScript output devices ranging from black-and-white laser printers, large format color proofers and laser film-to-plate imagesetters. The most modern plants opt to skip the film step and image directly to plate using very powerful thermal laser-imaging systems.

The most common network protocol is switched 100Base-T Ethernet, but some 10Base-T nodes still exist and gigabit Ethernet and fibre-based network protocols are becoming increasingly common as the file size and throughput requirements continue to grow unabated. The network server is vital for its ability to allow multiple operators to access portions of a project (known as a job) at the same time. Also, this server-based workflow allows for the job to be easily passed on to the next worker or team without requiring a huge amount of information to be copied to another location as it passes through the stages of typesetting, image scanning and retouching, design, proofing, imposition and finally PostScript output generation, ripping and imaging.

Figure 1. The Imposition and Output department. From left, Jimmy Carman, Jason Clifton and Doug Wilson flight check, impose and output to plate all sheetfed print work at Mahaffey's!. The tan machines with monitors atop at the left and right of the photo are Presstek thermal computer-to-plate imagers.

A recent 32-page color product catalog at Mahaffey's! exceeded 4GB of files when completed. It spent more than ten days on the server while being worked to completion. We average more than 800 print jobs per month, on a group of 17 offset and flexographic printing presses. Not all jobs are this large, but the need for a great deal of server space is a foregone conclusion in this business. It would be one thing if a fast, stable and scalable file server that could serve Macintosh and PC clients were all that was needed. However, these same machines are required to spool print jobs to the host of output devices that exist in the pre-press department.

The PostScript files they spool are typically imposed, multi-page, full-color printing forms and average 250 to 500MB each for a 32-page catalog consisting of eight 4-page forms. It is not unheard of for a single PostScript file to exceed 1GB in size. As you might imagine, a busy department with 10 or more employees working on and printing massive files to the server can choke a network in no time at all. The result might be slow print and serving times for all users, or a server crash—both cause expensive delays.

Several years ago, before the existence of the Windows NT server, the only choices for these machines were proprietary RISC UNIX servers such as Sun and SGI (Silicon Graphics, Inc.) or an AppleShare server from Apple. There was no middle ground. The high end was the high-priced domain of the big printing

plants costing \$75,000 and up, and the low end was low-powered and cost \$15,000 or less for small companies with only a few employees.

The Windows NT server broke into the pre-press arena around 1995/1996 with version 3.x and SFM-Services for Macintosh. This is a Microsoft implementation of AppleTalk file and print services and is much slower than the AppleShare/IP protocol common in UNIX and Linux implementations. Intel processors of that era were ill-equipped to handle the extreme duress brought to bear in this big-file and high-system-I/O environment, but the Alpha 21064 and later the 21164 chips from Digital Equipment Corporation proved equal to the task. These systems began to proliferate in the graphic arts industry.

The NT Alphas were not as scalable or stable as the UNIX machines, but they cost less, occupying the \$20,000 to \$50,000 middle ground. They were assumed to be easier to administer as well. Whether this was true is a subject worthy of debate.

Figure 2. The operator console of the five color Heidelberg MOF offset press shows a process color newsletter being produced for furniture manufacturer Lane. This two-sided 17x22 product exceeded 500MB in size.

In an OPI workflow, the imposition station (imposition is another vertical-market application that takes single pages and assembles them into press-ready flats) forwards a thin version of PostScript, one devoid of all high-resolution graphics, to the OPI server where the PostScript comments, instructions for placing high-resolution data, will be handled by the OPI application. The OPI leaders in the UNIX space are EtherShare OPI from Helios, a German firm with a long list of OEMs as customers, and FullPress from Xinet of Berkeley, California. The NT platform has several players with the largest and best-known being Color Central from Luminous.

We installed an NT Alpha 533MHz server with 256MB RAM and 60GB UW SCSI RAID in mid-1997, opting for Color Central for OPI serving. We also installed 3Com 10-100 switches and benchmarked our PowerMac clients at 2.3 to 2.6MB/sec file reads from the new server. This was almost triple the 900KB/sec we got over thin Ethernet to our AppleShare 4.2 server before the upgrade. We were pleased with the improvement. Around the same time, we began to experiment with Red Hat Linux version 4.0. The idea was to use Linux to determine if a UNIX system would be something we could administer ourselves. After struggling a bit with early installs on Alpha hardware, learning and working with Linux began to get easier and each new release seemed to make big gains in hardware support and usability. We implemented the first Linux server on our work LAN in late 1997 and used it for FTP and web serving at first. Later, we added DNS and proxy-server duty. We installed Netatalk and

Samba and configured the little homemade \$600 Pentium 166MHz 64MB machine to serve files to the Macintosh clients and talk to the NT server.

It was a good network citizen, and served files to and from our T1 Internet connection. We considered it a toy compared to the much more powerful Alpha hardware and PowerPC Mac workstations, but it was very stable and easily did its job. One day, while copying a large customer file that had been transferred via FTP, I observed an unusually short copy time. The file was around 25MB and it copied incredibly quickly to my workstation via the Macintosh finder. I copied it again and observed 7.5MB/sec throughput. I did this on two other machines and they all showed comparable speeds. It was unbelievable. This little toy that didn't even have SCSI drives was delivering three times the file transfer speed of \$20,000 worth of Alpha Iron with Ultra-Wide Barracuda RAID.

The secret, of course, was Adrian Sun's AppleShare/IP Netatalk patches. That very day, I began questioning why Linux wasn't used in the network server role in pre-press operations. One reason was low industry awareness, but the biggest single obstacle was lack of an OPI package. We are past the volume at which we can operate efficiently without OPI services. This lack of support was a show stopper. I began polling industry newsgroups and mailing lists more than a year ago, and noticed a dismally low number of folks in the trade had any awareness or interest in Linux. I found a couple of people using it for Internet duty and several more for simple file sharing, but not one was using it as a primary network server in a high-volume printing plant.

I began to lobby the folks who I thought were best positioned to port to Linux, namely the UNIX OPI vendors Helios and Xinet. At first, they ignored my public pleas for a Linux version. Then I actually sparred on-line with an Xinet representative about whether or not Linux was a viable solution in this role. A representative for AGFA, a European industry heavyweight, claimed to have worldwide market research showing that their customers wanted only NT for servers and RIPs. The only country with a high interest in UNIX products was Germany. He stated that Linux wasn't even on their radar screens during the last survey. The entire industry was dismissive about Linux in the graphic arts at that time, despite the growing interest and market-share numbers I was reading. Linux was starting to look good to many people, and I knew with a little nudge it could display its ability to vastly outperform NT servers in stability and file sharing speed at a fraction of the price.

Figure 3. The Heidelberg Quickmaster DI is the most automated printing press in the world. It runs four-color process and mounts, images and cleans its own waterless offset plates while it presets ink keys for the operator.

We installed two more Linux machines, both SMP Pentium Pro 200MHz with UW SCSI, and set about using Linux in a meaningful role in pre-press. We were able to share files, but when it came to printing, we had to defer to the NT machine because of the OPI support. We were successful in setting up a “back door” OPI workflow where we used the Linux machine to serve files, but OPI spooled to the Alpha. Color Central made file calls to the Linux server and forwarded the PostScript files to the RIPs. This gave us the benefit of faster file access on the Macintosh clients, but the penalty was much higher network traffic. This was more of an experiment to see if we could do it than a real solution. The only way to make this work without beating up the network would be to set up a private Ethernet channel between the two servers, but I had no open slots on the Alpha and would be robbing a chunk of the aggregate I/O on this server as well—not a good solution.

The first positive sign that someone was listening came when I was contacted off-line by Tom Hallinan of European MikroGraf Corporation (<http://www.ugraf.com/>), the North American distributor for Helios. He said he was gauging industry interest in a Linux port on behalf of Helios. I went on at length about how great it would be, but didn't get any kind of commitment. One good thing I learned was that Helios was about as steeped in UNIX experience as you could get—running on a stunning nine UNIX platforms. I was told the main issue was not the technical end, as their engineering staff had more than a couple of Linux fans on it, but whether the product had a commercial market on Linux.

Over the next several months, I didn't miss an opportunity to request Linux versions of printing industry applications. As the computer trade press started giving more coverage to Linux and eventually the mainstream press, I received an increasing number of requests from interested end users. I had also drawn the ire of several industry vendors who had thrown their weight behind Windows NT or seemed blindly tied to other platforms like AIX. One marketing representative from a very significant industry player, when discussing Linux growth in popularity, stated: “That's all we need—another platform to support.” I won't be doing any business with them until they change that attitude. This is the split I see as a Linux advocate in the graphic arts. I am getting an enormous number of requests for information from end users and industry associations—almost all positive. I am considered a pariah by the vendors who haven't figured out Linux' potential or seen its growth curve. For at least a year, I have been telling folks in my industry that this is big.

I attended the Seybold Publishing trade show in Boston this past March and directly lobbied Helios and Xinet representatives for Linux support. Helios was tight-lipped; Xinet was dismissive. I was rather disappointed, but ventured over to the Adobe booth and started asking about a port of Acrobat Distiller, another

tool we could use on Linux. I was directed to a technical marketing representative who said he used Linux at home. He had asked the developers for that very application himself, because he used it constantly and had to go to Macintosh or Windows machines when he needed it. I recounted my talks with the UNIX vendors, and he said, "an employee of Helios told me they already had it running in the lab." Wow, what a gem. This is the kind of stuff one developer might tell another, but not share with any end users. I left convinced that Helios would deliver the goods.

Two weeks later, they announced the port of their entire suite of high-end printing tools—the whole package. I actually found out from a Slashdot posting. I doubt that this was a big deal to the average user, but it was big to me. The last major obstacle to deploying Linux as a network OPI server had been removed. I contacted Helios and talked to Jim Davison who answered several questions, including the prices of the software I would need. He also pledged to give me access to their FTP server (Linux-based) when they got the Helios binaries, if I would like to test the software before the new CDs were pressed.

Figure 4. The Network Server and SeeColor Proofing RIP at Mahaffey's are shown here. A digital workflow requires an accurate digital proof. An Imation Rainbow Proofer at left has been nearly idled by the SeeColor-driven HP2000CP at the left. Linux could idle the NT Departmental Server in the near future.

This happened about a week later. They sent us temporary keys for 30 days of use and I downloaded the gzipped binaries from their server. I intended to test the performance of Helios Ethershare AppleShare/IP file sharing, and stated so on a computer-to-plate mail list. Xinet, not wanting to be left out, and several end users encouraged me to expand the testing beyond just EtherShare on Linux.

We decided to test Apple's new OS X Server, a BSD UNIX-based server product, since we could do a head-to-head comparison of Xinet and Helios on that platform. We had just received a copy and had two blue and white G3 Macintoshes that were able to run the server OS. I was also encouraged to add x86 Solaris to the test, since we would be doing the server test on an SMP P-II 333MHz IBM Intellistation M Pro. We had to throw the NT Server into the mix, as it represented what we were using and would show what kind of improvement in throughput we might expect.

I was contacted by a representative from Intergraph Computer in Huntsville, Alabama, who said they had a product about to go into beta testing that would allow Windows NT to perform AppleShare/IP serving. I told him I was interested and later signed a non-disclosure agreement before receiving the software for testing. The stage was set.

The decision to perform these tests off the pre-press LAN was based on the sheer number of operating systems and network services to install, and the fact that it was likely to disrupt work if done there. We decided to run the tests on a smaller test LAN with no PostScript printers. Since our test machines do not have the Ultra-Wide SCSI RAID that would be found in any normal pre-press installation, we decided the first round of tests would be limited to file sharing throughput speeds.

We attempted to keep the variables limited just to OS and AppleShare/IP stack, but the Mac OS X server runs only on Apple G3 hardware so those tests were run on a G3 300. We ran all the x86-based software tests on the same IBM hardware.

The primary test box running NT server, Solaris 7 and Linux is a dual P-II 333 IBM Intellistation M Pro with 192MB RAM and a 6GB Ultra ATA drive. The machine has UW SCSI on board, but this was not used in any of the testing. In the case of the Macintosh OS X server, the machine was a G3 300 with 192MB RAM and a 6GB Ultra ATA drive.

The Macintosh client used was a G3 350 with 128MB RAM and a 6GB Ultra ATA drive. The onboard 100Base-T was used on the Macintosh client and a 3Com 3C905TX was used on the IBM box. The network is running on a generic 8-port 100Base-T unswitched hub.

The test application was Helios LanTest 2.01. This is a test suite that runs the server through a grueling series of activities designed to simulate vigorous Macintosh client use. The test settings far exceed normal client activity for any user short of a maniac. LanTest first creates 100 20KB files, then opens and closes 100 files, removes 100 files, writes 3000KB to file, reads 3000KB from file, locks and unlocks the file 4000 times, reads a directory of 320 files and finally prints a 3000KB file. The default test settings of 3000KB reads and writes with printing disabled were used. This is a small file size for graphic arts, but was used so that the servers could cache the reads and writes if they were capable. Thus, disk hit delays would not be brought into the picture, since the drives in all the test machines are inferior to UW SCSI RAIDs that would be used in a real network server install.

Figure 5. This Rack houses three computers. Two are Harlequin RIP that drive the Presstek platesetters and Quickmaster DI. The third is an SMP Intel machine running SuSE Linux 6.1 that houses Quality's web site (<http://www.qualityprinting.com/>), e-mail, FTP, DNS and proxy server for the company LAN. The network switches, T1 and DSL routers are seen behind the monitor. Pre-Press Manager Jason Clifton is shown at right.

The goal was simply to hammer the network cards and wire and see how the OS handled the hammering and how fast. What I found fascinating was how widely the results varied on the *exact same* hardware when the only things changed were the OS and Appleshare stack.

When I reveal that SuSE Linux 6.0 with kernel 2.2.5 and Helios EtherShare was the top performer, I hope the readers of this article will realize the significance of this announcement. Eric Morris, one of the network engineers in my Linux Users Group (<http://www.lugoj.org/>), and I expected Solaris 7 to beat Linux—but it did not.

I was surprised at how much the test taxed the processors on Solaris/Xinet. It appeared to me that the reads and writes were not fully cached; the result was very inconsistent test data. The numbers were quite different each time I ran them, covering the range from 4000KB/sec to 8700KB/sec for reads. The test suite runs ten times consecutively, then shows an average. The number I am representing is this average. I actually got spikes of 9000KB/sec or better on all the UNIX flavors, but none was able to average that throughput over all ten iterations of the test.

When I reveal that Windows NT Server 4.0 SP4 native SFM was the slowest performer, surely no one is surprised. NT, like Linux, cached the reads and writes and took the load in stride. The processors never broke a sweat and no significant disk activity was observed. A single client, no matter how vigorous the activity level, was light work for these operating systems.

The same hardware/software was tested with Intergraph's ExtremeZ-IP, but as I stated earlier, I have signed an NDA and therefore cannot publish the results.

Test Numbers

Ironically, things have become even more interesting since I posted these results to three mailing lists. I have gotten all kinds of positive feedback from end users. Two people from Xinet contacted me regarding the original poor test results on the OS X Server. It seems this resulted from the Xinet test running in AppleTalk mode with IP support disabled. I agreed to rerun the test, and the numbers above reflect the new test.

An Apple engineer e-mailed to ask why I had chosen not to run the tests with their faster G3 400MHz server model. Because we didn't have one to run the tests on, I replied. He arranged to fix that with a loaner we can use for three weeks.

I have been contacted by another vendor about testing their hardware with Linux, but was asked not to name them. Access to this true server hardware should enable us to step up the next round of testing to the work LAN and include real-world benchmarks such as multi-client testing and OPI PostScript file creation.

At this point in our testing, Linux is “King of the LAN” and is looking like an excellent choice for file and print servers in a graphic arts firm.



Jeff Wall is Production Manager of Mahaffey's Quality Printing in Jackson, MS. He helped found the Linux Users Group of Jackson “because they didn't have one”. Happily married, he has a yellow Labrador dog and entirely too many computers. He'll discuss his Linux performance testing at the drop of a hat at jefferson1@speedy.linuxman.net.

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

[Advanced search](#)

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

Flight Simulators

Roman Melnyk

Issue #64, August 1999

A look at Linux in the Aerospace Training Industry.

Have you ever wondered how full-flight simulators and CATS (computer-aided training systems, Figure 1) are made? The answer, as one might expect, is that they are made with a lot of high-tech hardware and a million lines of code. It is a true marvel of technology where hardware and software meet to produce some of the most incredibly realistic simulations in the world today. Full-flight simulators are based on three principles: visuals, sound and motion. Using these three basic concepts, we are able to produce the most sophisticated flight simulators in the world. A full-flight simulator is an invaluable training tool for helping a pilot acquire the necessary skills and abilities to react to unpredictable situations.

Full-flight simulators (Figure 2) are built with a complex and effective six-degrees-of-freedom motion system. This motion system is responsible for producing the realistic feeling of takeoff, landing and in-flight turbulence. The visual projectors generate the real-world images as visual references. The heart of the entire system is the host computer, which is responsible for interacting with all the avionics hardware, motion and visual systems as well as simulating the aircraft's aerodynamics. CATS are a different breed of aircraft simulation. Instead of defining a hardware interface to interact with all the aircraft panels, CATS produce a graphical representation of all the aircraft panels (Figure 3), schematics (Figure 4), Line Replaceable Units (LRUs, Figure 5), and black boxes as they are sometimes called (Figure 6). By contrast, CATS provide a quite useful tool for training aircraft maintenance personnel involved, as well as pilots. They are also invaluable as a diagnostic tool for maintenance personnel in troubleshooting and diagnosing aircraft system failures. CATS are available in three configurations: single-screen (mini-CATS, Figure 7), three-screen (CATS, Figure 1), and seven-screen (CATS System Trainer, Figure 8). This last one is a full graphical representation of the aircraft cockpit with touch-sensitive screens.

This system provides the entry-level pilot and/or aircraft maintenance personnel an opportunity to master his cockpit procedural skills.

Figure 1. Computer-Aided Training Systems

What kinds of software and computing power are required to produce the realism required for flight simulation? Until not too long ago, this field has been the playground for big players in the industry such as IBM, Digital VAX/VMS and Gould. Today, with powerful CPUs being commonplace and mainframes almost obsolete, Intel has just about dominated the marketplace.

The company I work for (CAE Electronics Inc., Montréal, Quebec, Canada) employs 4000 people, half of whom are software engineers in the simulation hardware and software domain. Speaking objectively, I am proud to say that CAE Electronics builds the most technologically advanced full-flight simulators in the world today. It is up to CAE engineers to come up with new technologies in software simulation, both with respect to the CPU hardware and the OS required to produce quality products for our customers. Linux can be used as a dependable and reliable platform for commercial flight simulation software in the airline industry.

Figure 2. Full-Flight Simulator

A Little History

CAE has always been a company that believed in building its own software tools and basic software development environment. For the last nine years, CAE has been an IBM AIX RS/6000 house. IBM has been our foundation for building a stable platform under AIX to run thousands of software modules in a real-time environment, a tough task for Linux to undertake. In the last four years, however, Linux has been used as a basic development platform for CAE engineers requiring a complete simulation environment.

Figure 3. Aircraft Panels Representation

Can Linux be used to develop CAE commercial software simulation systems? Yes. Linux has matured a great deal in the last couple of years. With that in mind, CAE engineering has been instrumental in supporting Linux. Will our customers react favorably to Linux? Will the Linux community support the product in the years to come? Both are serious questions. Full-flight simulators are a long-term investment for our customers. These and many other questions must be answered before we can fully commit and support future Linux development. Current industry trends show Linux being supported more and more by companies such as IBM, H-P, Compaq and Dell. The future for Linux looks bright.

Our basic principles in software simulation are simple. We produce C code in straight-line execution within a CAE software environment called SIMEX (simulator executable). This environment provides the basic software dispatcher required in controlling software module execution in a real-time environment. SIMEX also provides software configuration control under RCS to maintain multiple configurations for testing and integration. SIMEX is capable of building different executables for runtime execution from a collection of different source modules. This provides an invaluable environment for the engineers to test new software code without destroying the previous working configuration.

Figure 4. Schematics

Since AIX is a non-real-time OS, the question arises as to how one would create a real-time environment on a non-real-time OS. The answer is simple: produce your own library of asynchronous I/O control functions that the software engineer will require for all I/O system calls. Setting the system priority of the software dispatcher code to the highest level is not enough to create a real-time environment. All simulation I/O requests must be handled in an asynchronous manner in order to maintain a real-time simulation environment. We call this environment CAELIB (CAE Library).

The software dispatcher is also required to control the sequence and duration of software module execution so as not to allow any overruns to occur. This technique is called banding. Software banding enables the software dispatcher to control the time required to execute a collection of software modules as well as the sequence of code executions. Certain modules need to run before others in order to maintain aircraft system simulation integrity. This allows all modules ample opportunity to execute in the allotted time. For real-time simulation, this is 60 Hertz or 16.66 milliseconds for one full pass for all software modules. The software dispatcher then further breaks the banding down into legs, which are executed in a controlled state.

Figure 5. Line Replaceable Units

Not all modules need to be executed at a rate of 60 Hertz. Most aircraft systems need to be executed at only a third or even a quarter of a second. For this reason, the software dispatcher calls certain modules at an iteration rate of 16.66, 33, 66, 133 or 266 milliseconds. The CAE engineer enters his/her module into a dispatch table. This, in turn, instructs the software dispatcher code to execute and sequence as per the engineer's request. If there are too many modules in one particular leg and the code is unable to complete execution within the allotted time, the software dispatcher will terminate the execution of the code and continue onto the next leg for execution.

This situation is called a software overrun. In order to control software overruns, the integration specialist and software engineer redistribute the software modules into less heavily weighted bands that have excess execution time. This allows critical modules ample time to execute, while non-critical modules are executed at a slower iteration rate. Since all our code is based on legacy FORTRAN 77 and ANSI C, porting our code to a Linux PC platform proved to be a relatively straightforward task, except, of course, accounting for the obvious differences between Linux and AIX compilers. (More to follow on that subject.)

Figure 6. Black Boxes

The Task at Hand

The software simulation configuration discussed in this article was a Bombardier/Canadair Regional Jet RJ-200 aircraft (Figure 9), a successful 50-seat twin jet regional aircraft manufactured and built by Bombardier Aerospace in Montréal and popular with airlines such as Comair, Skywest and Air Canada, just to name a few. The first task at hand for CAE engineers was to port the entire CAELIB and SIMEX software environment to Linux. This proved to be a relatively uncomplicated task for our operating system support group (OSU), since most of our code is in ANSI C. All modules were compiled with relatively minor modifications. Once this was done, the task of transferring and porting the entire software simulation code from SIMEX to Linux proved to be more challenging. Now, I am referring to years of software development on AIX with at least 500 software modules, each ranging from 1,000 to over 100,000 lines of code—the porting of which would be no trivial matter. The third task was to port the required CAE-specific compilers from our own OpenGL-based graphics compiler and editor to our basic support utilities. CAE has developed its own graphical environment called TIGERS (an OpenGL-based system).

Figure 7. Mini-CATS

At the time TIGERS was developed at CAE, sophisticated graphic systems for simulation use were not yet commonplace. Figures 10, 11, 12, 13 and 14 show the level of graphical detail required for CATS simulation. The main difference between a CATS and a full-flight simulator is simple: CATS requires that all aircraft panels be a graphical representation, as opposed to the hardware-driven interface on a full-flight simulator. Simply put, a graphical representation of a switch is used instead of a real switch in a cockpit environment. In all respects, the amount of simulation software required is much more complex on a CATS as opposed to a real full-flight simulator. All aircraft systems are simulated in the CATS environment, instead of only the simulated black boxes of the full-flight simulator. In flight simulators, we simulate the same black boxes as used on the aircraft. In trying to address the needs of a Bombardier's

training department as they sought a portable version of their existing AIX CATS on the PC platform, CAE engineering decided Linux would be the ideal and practical choice for this project.

Figure 8. Seven-Screen CATS System Trainer

We made the decision to use the standard GNU ANSI C compiler and F77 translator for the task. Our FORTRAN77 legacy software modules proved to be the greatest challenge. AIX FORTRAN77 compilers are very forgiving in equating logical to integer data types. For example, if you equate an integer label to a logical label, the AIX FORTRAN compiler would assume you require the value in the logical label to be stored into the integer label. This style of programming is not allowed under F77 Linux. You must declare and equate labels of the same data type, or the compilation will fail. Now, this would be a simple task if the labels were defined in the local module. The problem arises when the software module in question uses a logical label declared in what we call our CDB (common data block) as an integer label. This common data block is used to declare all global labels used by every simulation module to pass variables among each other. If the declaration is wrong in the CDB, a correction must be made in all modules. Then each module must be reprocessed with the new CDB and recompiled.

Figure 9. Bombardier/Canadair Regional Jet RJ-200

The next problem lies with the memory storage format of logical labels between AIX and Linux. For example, a data type of Logical*4 (four bytes) in F77 Linux is stored from left to right (MSB, most significant bit, to LSB, least significant bit) while AIX stores the logical in reverse order (LSB to MSB). This can create a major problem when using the logical in a bit manipulation operation. If you use each bit for a different function in your code and apply a bit manipulation operation (e.g., **FECDE .XOR. Logical Label**), then the problems are compounded. You must reverse all masks used in the bit manipulation operation—a considerable amount of work.

Figure 10. CATS Simulation Detail

Another area of concern is the data files used to store navigation and engine data interpolation information. The byte order in these data files must also be reversed to read data correctly into their assigned CDB labels. Our ANSI C code does not have this problem. Most of the C code compiled with relatively little change.

Figure 11. CATS Simulation Detail

Aural warning system conversation required a minor amount of work. Under AIX, all audio warnings were designed to work with a DSP card with one large digital audio file. We were required to convert the large digital aural warnings data files into separate PCM wave files. A newly coded audio software module was then needed to cross-index the requested aural warnings to their corresponding wave file and pipe them to a standard SoundBlaster audio card. Throw in all the graphics files and IOS pages (Instructor Operator System Pages, Figure 15), and we have an enormous porting task.

Figure 12. CATS Simulation Detail

Success in Sight

Linux and CATS proved to be a good match. System reliability and performance met and exceeded our expectations with amazingly predictable results. CAE now has a full-blown aircraft simulation, with graphics and aural warnings, running in a Linux environment. Our Linux development environment consisted of a Pentium II Intel 350MHz CPU with 128MB of memory and a Toshiba 233 MMX Portege 320CT laptop with 32MB memory. As described previously, the CATS comes in three flavors: one-, three- and seven-screen versions. Under AIX CATS, the IBM host drives three separate X workstations, with each screen displaying different information. Under a single-screen configuration, one X workstation provides functionality for all graphics pages while using a cockpit navigational panel as shown in Figure 16. The top right-hand quick-reference panel is a small graphical representation of the aircraft cockpit. When one of the panels is selected, the system displays the selected panel in a designated portion of the single screen, with the seven-screen CATS being displayed on seven separate X workstations. Each displays a full graphical representation of all the aircraft panels (Forward, Overhead, Pedestal and Side panels, Figure 8). With Linux, the hardware configuration is a bit different. The graphics display is the same (one-, three- or seven-screen configurations); the difference is in how and where we display the graphical pages. Under Linux, we use a graphics card called Evolution 4.

Figure 13. CATS Simulation Detail

This card is manufactured by Color Graphics Systems in the UK. It allows us to display four simultaneous X display outputs using Xi Graphics' Multi-Headed X server. AcceleratedX is an excellent product, allowing us to drive up to 16 X displays (using four Evolution 4 Video cards) simultaneously. The hardware cost is reduced significantly when compared to three IBM X workstations with an RS6000 IBM host CPU and a Linux PC with one Evolution 4 video card.

Figure 14. CATS Simulation Detail

In a MINI CATS version (single-screen configuration), we now have the option to install the CATS on a standard PC laptop using the Xfree86 X server included with every Linux distribution. Distribution and installation of CATS has also been addressed. Customers did not want to install a dedicated Linux system on a previously preloaded Microsoft operating system. So, we developed a CD-ROM that would boot into a Linux environment, set up a small virtual Linux file system in memory, and loaded all the required CAE CATS software without installing any software on the local hard disk. This also has proven to be successful. Bombardier can now distribute their CATS systems on a pre-configured bootable CD-ROM. Booting and loading CATS in the field now requires no previous knowledge of or experience in the Linux operating system—no more file corruption or accidental file deletions by the customer. The primary reason for developing PC-based CATS for Bombardier was cost and portability, portability being Bombardier's top priority for their training instructors in the field. All these factors translate into a more affordable CATS for Bombardier and their customers.

Figure 15. Instructor Operator System Pages

Figure 16. Cockpit Navigational Panel

Linux has proven without a doubt that it is a reliable and stable development platform for delivering commercial-grade simulation products to the airline training industry. CAE has recognized the potential in Linux and its potential in future CAE simulation products. CAE may even explore the possibility of running a full-flight simulator on Linux one day. Job well-done, Linus Torvalds.



Roman Melnyk works as a Software Integration Specialist /Software Engineer for CAE Electronics, Inc. He can be reached at romanm@cae.ca.

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

[Advanced search](#)

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

[Advanced search](#)

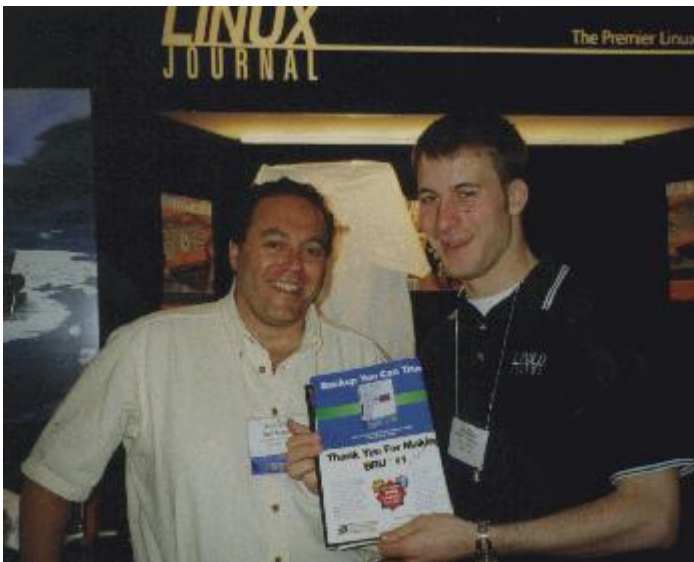
Linux Expo 1999

Marjorie Richardson

Issue #64, August 1999

Linux Journal attends Linux Expo in Raleigh, North Carolina.

Red Hat proved once again that they can put on a good show for the Linux community. Bigger and better than ever, Linux Expo again doubled in size and attracted top speakers such as Dr. Peter Braam of Carnegie Mellon University) and Dr. Theodore Ts'o, of MIT, who now works for VA Linux Systems. Big business was there too, represented by such companies as IBM, Hewlett Packard and SGI (formerly Silicon Graphics), as well as the usual Linux vendors, such as SuSE, Caldera, VA Linux Systems, Enhanced Software Technologies, Cygnus and many others.



Ted Cook of EST and Matthew Cunningham at the *Linux Journal* Booth

I talked to Dave McAllister of SGI about their involvement in Linux and Open Source and found SGI to be much more committed to this community than I would have suspected. They released their most robust and scalable file system, XFS, to the community in an effort to aid Linux in reaching what he

called “Enterprise level”. This is certainly something that was applauded by everyone I talked to at the show.

One of the most exciting announcements before the show was O'Reilly's and HP's sourceXchange.com web site. I attended a discussion about this site, which is designed to aid in getting needed open source developed by obtaining sponsors who will pay developers to write the code they need and then release it to the public. This is an idea whose time has come, as another group has also started a web site for the same purpose—this one is CoSource.com from a couple of independents, Bernie Thompson and Norman Jacobowitz, who write for *LJ*. It's obvious that Bernie, Norman and O'Reilly are committed to the community and wish to drive open source development, but I was a bit suspicious of HP. When I asked about HP's motives for involvement in this project, Wayne Caccamo told me HP felt this project was inevitable and wanted to take a leadership role *and* they wanted to “ingratiate” themselves to the Open Source community—talk about honesty! After that remark, I was ready to believe anything. I'm looking forward to seeing how both these sites work out. (For more on this subject, see Bernie Thompson's article in this issue, “Market Making in the Bazaar”.)



[Alpha Processor, Inc. joins Linux International: Jon Hall, Richard Payne and Guy Ludden](#)

There were the usual fun things to do, such as a chili pepper sauce contest and a paintball contest pitting vi against Emacs once more. Once again vi won, proving it is the best editor available or that its advocates are the best shots. More than one group bought blocks of tickets to a local showing of *Star Wars—The Phantom Menace*. The ALS (Atlanta Linux Showcase) group invited me to go along with them. Fun movie.



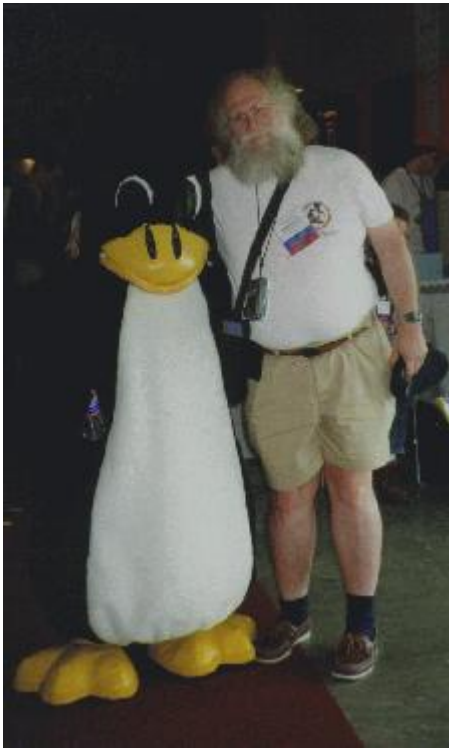
Dr. Peter Braam

I especially enjoyed my booth time talking to current and future readers and authors. In particular, it was a pleasure to finally meet Alan Cox and Telsa Gwynne.



Alan Cox and Marjorie Richardson—we meet at last.

Alpha Processor, Inc., a Samsung company, announced they were joining Linux International; Richard Payne and Guy Ludden presented a check to Jon “maddog” Hall. I got the picture and then took several others of Jon, including one with a people-size Tux, who was roaming the show floor.



Tux and Jon Hall

Compared to LinuxWorld, Linux Expo came across as more polished, more “we've done this before successfully”. LinuxWorld had a lot of glitz—electricity and energy filling the air—that just wasn't there at Linux Expo. I think this mostly had to do with the fact that it wasn't the first time for these guys—the experience showed. The speakers all like Linux Expo better, as the Expo paid their travel expenses while LinuxWorld left them to get there on their own. LinuxWorld had more people and more vendors, but they also have the advantage of being in Silicon Valley.



Linuxcare Labs Booth

Evan Leibowitz described the Expo as “the show where Linux lost its innocence” due to two unpleasant situations that arose. One was Pacific HiTech's being kicked out for passing out t-shirts without buying booth space. The other was the use of the Red Hat trademark without permission, by LinuxCare on their

poster parodying a Palm Pilot ad. No matter which side you took on these incidents—the calling of lawyers certainly signals the “end of innocence”.



[ALS helps out at the Linux International booth: Blake Sorensen, Hunter Eidsen, Greg Hankins and Jon Hall](#)

The show was definitely a success. I talked to Bob Young on the last day, and he certainly seemed pleased with how it had turned out. See my interview with Bob in this issue. For vendor announcements, see “UpFront”.

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

[Advanced search](#)

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

Advanced search

Interview: Robert Young of Red Hat Software

Marjorie Richardson

Issue #64, August 1999

Getting off the showroom floor turned out to be the real trick, as we were often stopped by people wanting to shake his hand, ask him a question or just tell him how wonderful Red Hat is.



I talked to Bob Young on Saturday, May 22 at Linux Expo. He met me at the *Linux Journal* booth, and we walked to a quiet place in a nearby plaza. Getting off the showroom floor turned out to be the real trick, as we were often stopped by people wanting to shake his hand, ask him a question or just tell him how wonderful Red Hat is. Even during the interview, we were “found” by one fan who sat quietly and listened until we finished and then jumped up to shake Bob's hand.

Margie: Do you have any numbers on the show?

Bob: No, I don't; you'll have to check with Lisa Sullivan. I actually just arrived at the show; I believe we are doing pretty well.

Our goal, of course, was to keep up our trend, which is to double the show every year. In 1995 at North Carolina State, we had about 200 people; the next year we had about 400 people, again at NC State. This actually wasn't “we” at

this point, but a couple of students. Red Hat sponsored it in 1997 at the Biotech Center, and about 900 people came. Last year, the show was held at Duke University and we had about 2000 attendees. So our goal for this year was to attract 4000 people to the show. As of Thursday noon (after the first day of the show), we were at 3000 people. Given that most of the attendees come for the first day of the show, that was probably the bulk. This number does not include vendors or speakers. I think we are well on track to making 4000, but we'll have to wait to actually see what the numbers are.

Margie: It certainly looks like 4000. [Final count was 3600.]

Bob: Oh, good! It's a great show. One of the things I've been noticing is everyone seems to be coming every day.

I've seen a good number of people in the exhibit hall on multiple days—normally, people do the exhibits in the morning, then are done—just coming back to the conferences. But these people are spending a great deal of time with each and every booth.

Margie: That's good to hear. Are you going to keep the goal of doubling each year?

Bob: Absolutely—if not faster! I love to quote Linus' line whenever anyone asks him about the future of Linux: “World domination and fast!” While we may be pleased with the success of 4000 people, it isn't quite world domination.



Margie: 8000 next year—good luck on making it. There have been a couple of incidents that people are talking about. One is the Pacific HiTech t-shirt giveaway. Giving out t-shirts is not allowed? I heard they were run off.

Bob: Yeah, this is just show management issues. You'd have to check with Stacy, who is running the show, for more detail. Typically, for vendors to attend a trade show, they buy booths. Vendors who try to sneak into a trade show without buying a booth in order to get the benefit of the trade show—well, whether you go to COMDEX or any of the other shows, you'll be run off pretty

quickly if you try to get the benefits of the trade show without contributing to the cost of putting it on.

Margie: That makes sense—how about the poster for LinuxCare?

Bob: That's a funny story, but one we take very seriously. In the end, the only thing Red Hat owns are our trademarks—we don't own any intellectual property—so we have to defend our trademarks very carefully because we can't afford to have them end up in the public domain. Legally, if you don't protect your trademark, you can lose it. We were very surprised by LinuxCare's cavalier attitude. It's one thing to do a parody of someone's advertisement, which the poster was—a parody of the Palm Pilot ad. But it's sort of odd to use a third company's product in that parody. If LinuxCare had put a LinuxCare logo in the woman's hands, that would have been different. But to use our product without our permission as a parody of a Palm Pilot advertisement—well, we were just very surprised they didn't talk to us about it beforehand.

We have a second problem in that Linux users are 99 percent male. The Linux community has not been perceived as a very welcoming community to the other 50 percent of the population. To do a parody that takes an artistic nude from the Palm Pilot ad and turns it into a sexual nude just makes it seem we are going in the wrong direction.

Margie: I truly agree with you on that one. I went by the LinuxCare booth before y'all asked them to remove it. I saw it and told them I found it offensive.

Bob: Well, thank you. It was Lisa Sullivan, our marketing director, who saw it and said, "What are these people thinking?" She went over privately to ask them to stop passing it out because it was sufficiently offensive and tacky, and on top of everything else, it infringed on other people's trademarks. They chose to ask her to talk to their lawyers instead of just complying with the request—so we did. And that's where the story came from.

Margie: That's amazing—I do hate seeing lawyers called. Matthew Cunningham, my cohort here, took a tour of your building and was very impressed. He wanted me to ask you how you decided to use cubicles instead of private offices.



Mr. Young's cubicle, where it all happens

Bob: We actually used private offices in the past. We started with a very open plan; mind you, this was back when we had no money. We couldn't afford cubicles or offices; we just had folding tables. When we moved into our first facility in December 1997, we decided it would be much more efficient if we could close doors and concentrate on the work at hand, so that's the route we went. After working in that environment for a year, we recognized something: while there is a downside to both models, the open space has the one advantage of enabling communication. It's so critically important in our business—in any business that's evolving as quickly as both Red Hat and the whole Open Source movement. We are willing to put up with the distraction of interruptions and noise and everything else you get in an open space in order to focus on the primary benefit, which is the instantaneous communication you can get if you don't have offices.

Margie: Matthew was particularly impressed that you have a cubicle yourself.

Bob: I'm the one who gets cut off the fastest from the information flow around the business. I think it is doubly important that I'm sitting in a cubicle!

Margie: What's going on with 6.0? What's new?

Bob: There are many new things in 6.0. My job isn't so much to make products anymore or come to trade shows and sell them—my job is worrying about the health of the business five years from now. That's why 6.0, as far as I'm concerned, is so important. When you look at the health of Red Hat, you have to worry about the health of the Open Source movement in general.

With 6.0, and I presume it's true for Caldera's and SuSE's and Debian's and Slackware's builds of the Linux operating system, the acceleration in development of this Open Source community had to be considered. What's been happening is we've been getting teams from SAP doing Linux kernel work because they want a faster, better OS for running their application engine. Just recently—this hasn't had an impact in any Linux products yet—SGI released

their XFS journaling file system, which is a world-class piece of technology, as Open Source. This is indicative not only of the fact that the open-source development model is working, but also that it is actually accelerating.

In the 2.2 kernel, we are seeing things like the Linux Graphical Network Management tools that compete very effectively with the best management tools you'll find in any of the other UNIX systems, or for that matter, in NT. Of course, we are also getting all the client-side tools, now that we've got both the GNOME and KDE projects rapidly moving forward. The GTK toolkit has become a standard. Netscape reporting all Mozilla code means moving Navigator and Communicator over from Motif to the GTK library. Of course, we already have the GIMP for which GTK was named, and AbiSource is building their word processors against GTK. The GTK libraries, which are under the LGPL, are very exciting because if the client is going to succeed, we have to be able to support graphical applications.

Margie: Each year, Red Hat has gotten easier to install. Now Caldera has their new Lizard install that's really just insert the CD and have it do all the work. Have y'all put in the same sort of easy install that they have?

Bob: We are very impressed with Caldera's Lizard-thing—it's very slick. It has received some very good reviews. We do have certain concerns with our install in terms of going in that direction. Among others, there is always going to be some hardware that is not supported, so you need to be able to back up when you run into a problem. If we build a graphical front end, we want to make sure that during the installation process, you can back up. That aside, you can be assured we are working on a graphical front end to the installation tool set. We also want an open-source one.

We are very interested in what Bodo Bauer is doing with his open-source install shield for his new company, Zenguin. That looks like a very interesting project. I think on the installation side, there is going to be a ton of very rapid advances. This is what is so powerful about this community approach. This is where Caldera is presenting huge value. If it wasn't for Caldera's Lizard install, we might go off and work on something else instead. But with someone pushing us, we're going to have to leapfrog back and come up with a new set of technology. It's that competition in the marketplace that's causing the Linux open-source-based technology to innovate faster than you've seen in the proprietary OS technologies coming out of Microsoft—no one is pushing Microsoft.

Margie: Well, hopefully Linux is pushing them now!

Bob: I suspect you'll see a little more innovation out of Microsoft now.

Margie: Okay, Red Hat has all the big guys giving them venture capital. How did you manage that? Did you go looking for them?

Bob: Yes, we refer to it as “Elephant Hunting”. We went looking for them for a particular reason. In January of 1998, we won InfoWorld's product of the year award for the second year in a row. The first year, we tied with NT; in 1998, they gave it to us outright. We could have been flattered and sat back and said, “Wow, that's pretty cool.” Our reaction was “Houston, we have a problem.” The problem was this: Linux was winning awards everywhere—SuSE in Germany, us in many other marketplaces with many other journalists, and the Apache technology was winning—yet the MIS directors we were talking to were choosing NT anyway, because the MIS directors are looking five years ahead. When they adopt new technology, they have to worry about the investment they have to make in that technology in terms of training and implementation, and they have to worry about getting a five-year life out of it. With the open-source technologies coming from the small companies, they are saying, “Let's see, I can work with a bunch of guys down in the tobacco fields in North Carolina or I can work with the world's most profitable software company, that being Microsoft” and making the easy decision.



The problem is, even if Red Hat doubled in size or even became ten times bigger, we would still be way too small for an MIS director at Ford Motor Company to trust our technology. So we realized the only way around that problem was to get the endorsement of the industry leading players. So that by first getting into Netscape and more recently getting Compaq, Dell and IBM involved, you now have people like Michael Prince at Burlington Coat Factory deploying 1200 Linux boxes he got from Dell, because he could rely on Dell five years from now. So while Red Hat may not be around in five years, clearly Dell Corporation will be.

That was the whole point to it, getting the big ISVs—the Oracles and SAPS on board. Netscape indicated to the ISV community that this was serious technology, and despite the fact that it looks more like the Internet in terms of not being a controlled technology, it nonetheless is a very powerful technology

model and has very effective means for ISVs to deploy their technology on these operating systems that are being built on the Linux kernel.

As a result, it's all in the interest of sending the right message both to the ISVs (the industry) as well as to the users—that this stuff is ready to be used. I think we are beginning to see that, in terms of the interest of the industry and the world at large and what we are all trying to do.

Margie: How much are you participating in the Linux Standards Base?

Bob: We are hugely supportive of standards! If you go to the LSB mailing list, you'll find many Red Hat engineers very actively involved. When Dan Quinlan is quoted on this subject, his quotes always seems to have a big question mark in the middle—why are you asking me about this?—because he knows that Erik Troan and Christian Gafton and all our lead engineers are very actively working with him to build the LSB. Yet, we seem to be criticized in the press for not supporting the LSB.

There is only one place where I think we are possibly vulnerable to that criticism and it is unfair criticism. We come out of the Linux world, and one of the big attractions to the Linux world is that you publish all your source code—you publish early and you publish often, as Eric Raymond says. Thus, everything you do is in the public space; you are living in a fishbowl. You don't have the opportunity to do vaporware-type announcements, and we don't. So even when it comes to standards efforts, we don't want the world to think Linux is going to fail, unless a standards body like the LSB succeeds; in other words, what if, for whatever reasons—lack of support by other teams or a standard that is not implementable—it fails? Suddenly everybody writes off Linux, saying “Oh, they couldn't get their act together and do a standard.” We're interested in seeing that standard in place, but we're not willing to bet the future on it. We're going to work very hard, harder than anyone else to make sure of that standard; Red Hat has always adopted things like the Linux file system standards. Everything the Linux community tells us is going to become the standard. glibc was going to become the standard, so we adopted it very aggressively. Then we were getting criticized for adopting glibc, but we were thinking, “Hold on, the whole development community is telling us the future of the technology is glibc.” So you get into a lot of this kind of stuff where we are very focused on staying very close to the development communities. As such, we will not use the media to promote projects or technologies that are not yet real. Just because that's what Microsoft does, and people are so fed up with it that I don't think any of us should go there.



Margie: I agree; I was in programming long enough to know that vaporware isn't a good idea. When Standards do get put in place, how do you think distributions will be able to stay separate?

Bob: It depends on what you're trying to do with Standards. We think Dan Quinlan is going in exactly the right direction. This goes back to the Swiss stock exchange and the MIS director to whom I was talking. He asked me specifically, very pointedly, the question of how do you avoid the standards trap? That is, if you define your standards too narrowly, you limit innovation, because suddenly every player in the industry has to go back to a 30-member standards committee to ask permission to deploy another piece of technology. You must define your standard clearly so that it delivers specific benefits and defines the framework that allows the various Linux engineering teams to build better and better operating systems out of the kernel, and the X Window System and all the pieces that come out of this Open Source community. We define it broadly enough so that the developer is not limited in the innovation he brings to it; then we get the best of both worlds. If we define the Standard badly, so that it's not implementable because it doesn't enable the technology to evolve forward, then all we have successfully done is handed the future of the operating system space back to Microsoft.

So that's the balance and that's why we're saying we're going to support this—we want to see it done properly. But there is the risk, and Dan Quinlan will admit to it, that this thing might not work, and if it doesn't work, we will continue to work very closely through Linux International with “maddog” and everyone else to ensure communication keeps working so our users get the benefit. As I put it, using a car analogy, you need only a single driver's license to drive any kind of car from a BMW to a Hyundai. That is what we want to ensure—that you need only one driver's license to drive any version of Linux.

Margie: You have had a lot of partner announcements, and people have products coming out ported just to Red Hat Linux. Many people are afraid that

this is going to cause Red Hat to become the Microsoft of the Linux world. Do you think that is going to happen?

Bob: No. Here's why. We don't do any exclusive contracts. For example, the guys at Metrowerks—on their box, they say it was built for Red Hat Linux. That wasn't our idea—that was their idea for marketing purposes. They recognized, with the partnerships we've been making, that they wanted access to our marketing channels and to send a message to our users that this product was specifically targeted for them. I suspect they will do Code Warrior for Caldera and Pacific HiTech and any other distribution that achieves a decent market size. ISVs are porting to Linux now only because they perceive there are a lot of Linux users. For example, our goal at Red Hat has always been to expand the number of Linux users, not to dominate the Linux space. When we started, Slackware had half a million users—95 percent of Linux users at that time. Today, they have a much smaller market share, probably a million users. Slackware has in fact doubled their number of users in the period we've been around. So we haven't done a particularly good job at squashing our competition! And we have no interest in doing that. Slackware and Caldera and Debian are very much our allies in this effort to make Open Source the definition, to make Open Source a required feature of any operating system the user might consider. If we can do that, we have a real shot at displacing the proprietary binary-only operating systems, whether they are from Microsoft or Apple or other UNIX vendors or IBM.



Margie: I talked to Metrowerks on Thursday, and Mr. Bélanger said they are talking to other people.

Bob: Of course they would, because they would be foolish not to. It's as simple as running **make** to rebuild it, even if the market they are trying to build it for is, for example, using an older C library. Most of us are using the same C libraries these days, so the stuff probably runs. I don't know this for certain; you'll have to check with the Metrowerks guys, but I suspect it runs on the other distributions just fine.

Margie: They didn't tell me that, just that they would be talking to others. I was talking to Clay Claiborne, a Red Hat reseller, and he told me that Best Buy sells Red Hat for \$65 while he has to pay \$69.95 for it. How does that kind of pricing

work—that companies like Linux Mall and Frank Kasper are charged a higher price than Best Buy?

Bob: It's a real challenge for us. According to U.S. law, you are not allowed to have multiple pricing schemes other than for volume. So we are wrestling with this one as we speak. We have a lot of loyal resellers. The guys at the major shops are quite willing, if they have a product like an operating system—this used to happen with the Macintosh OS at places like MacWarehouse—to sell it at a loss because they know customers will also buy more memory and a hard drive and maybe even another computer. They are quite happy to sell that operating system at a loss in order to get all the additional hardware benefits from it. Also, the guys at Best Buy and Comp USA are competing with each other for retail customers. As far as I can tell, they are not making any money selling Red Hat, but they perceive it is in their best interest to market their products that way.

Margie: Are you saying that Best Buy is paying \$69 for it like Frank Kasper and Linux Mall, then selling it at \$65 as a loss leader?

Bob: I am the wrong guy to ask, to be perfectly honest. I don't even know what our pricing is! I'd have to send you over to someone on our marketing side to give you precise answers. But yes, that is a problem because they move huge volumes, and as such, they get big discounts and smaller resellers don't get quite the same discounts. All I can say on that point is that we are actively working on it. We have clearly heard the message from the channel and we are going to fix that issue as well as we can.

Margie: Caldera started out as the expensive distribution and has dropped their price, while Red Hat has gone up to \$80.

Bob: No, you see, there is complete fallacy in there—we give away our software for free, and we continue to give our software away for free. So our price has gone from zero dollars to zero dollars, and we are hard pressed to see how there is a pricing increase in there. What our price increase relates to is we do have a series of products based around the software you can get for free.

We understand our competition isn't with Caldera or SuSE—our competition is with Microsoft. If we are going to build world-class support organizations and deliver world-class documentation so that customers, who are used to buying products of the caliber Microsoft produces, are going to choose to work with Linux and Red Hat instead, we can't underprice that service. If we do, we won't generate enough money to be able to deliver that quality of service and support to our customers. So our flagship official Red Hat Linux box set is now an \$80 product because that's the minimum price we think we can charge to

provide world-class support and documentation in that product. That product price may in fact go up in the future if we conclude our customers are actually asking us for it.

Meanwhile, we do have an upgrade version available for \$40 on our FTP site called the Core Product which is the full version, and of course even here at the show, you've got Linux Central selling \$3.00 Red Hat 6.0 CD-ROMs. I'm not sure we can be accused of being the high-point guys; I think we are the low-cost vendors, but then so is Slackware and so is Debian. As far as we are concerned, we think the guys at Slackware and Debian have much bigger market shares. If we were trying to compete in the Linux space, we would be focused on Slackware and Debian. But we are not trying to compete in the Linux space, we are trying to compete with the very big proprietary OS vendors who have hundreds of millions of customers.

Margie: What can we expect to see from Red Hat in the future?

Bob: Well, hopefully more and more of the same. We think there are huge opportunities to do very big projects in an open-source model. We really like what the guys at AbiSource are doing with their products. We think the guys at Cygnus are going to be doing very clever projects in the open-source space. Of course, we are going to be doing even more projects along these lines.

The enthusiasm you see here at the show has little to do with any product, and everything to do with, for the first time, the user getting real control over the technology they are using. That's the one unique benefit we all deliver to the marketplace. We intend to do more of that and do it better as we generate the resources that enable us to do it. If the customers like it, we'll generate the resources—we're a very customer-focused company. Go talk to some attendees and find out what they want us to do, and that's probably what we are going to do!

Margie: There is a lot of speculation that Red Hat is going to go public. Is that going to happen?

Bob: No comment.

Margie: No comment? Not going to give me a scoop, eh? Well, that was the answer I expected, but I had to ask. [On June 4, Bob announced that Red Hat was going public (www.redhat.com/corp/press_ipo.html).] Thanks very much for talking to me.

Touring Red Hat

Red Hat office photos by Matthew Cunningham.

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

[Advanced search](#)

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

[Advanced search](#)

Market Making for the Bazaar

Bernie Thompson

Issue #64, August 1999

Mr. Thompson presents a bold and innovative business model for paying developers to produce free software.

In “Making Money in the Bazaar” (June 1999), we raced across the landscape of current efforts to drive innovation and make a living in the Open Source market. We now introduce a system for consumer-driven Open Source funding. If successful, it could accelerate the pace of innovation even further and create a small industry around developing free software.

The Need

Is there some graphics hardware you wish Linux supported better? A game you wish was ported from Windows? Or possibly a GUI application to ease some parts of system setup?

If so, what do you do about it? If you are a developer, you can just write it. The Linux model has been “scratch your own itch” and contribute back to the community.

If you aren't a developer, don't have the time or the necessary skill—you're out of luck. You have to wait and hope that some other sufficiently motivated developer with the same need will take on the project.

This can be quite frustrating—you need that graphics driver *now*. You could speed things up by hiring someone to develop the software just for you. But paying a fortune to have a custom driver developed for a \$50 graphics board just isn't feasible.

A larger group is needed to share the burden of developing a standard driver. Several thousand people in the world probably use that same graphics card

and run Linux. Why don't some of them get together and share the cost of having the work done?

The same concepts apply to scripts, help files, applications—anywhere there is a demand for software.

A Buyer's Co-op for Open Source

The Internet is an amazing tool for bringing specialized communities together. A group of people needing the same software is just such a community.

The trick is to attract all of the interested parties to the same web site, where they can pool their resources with others wanting the same thing. This site must coordinate the process of gathering support, selecting a developer, evaluating the resulting software and collecting the funds to pay the developer.

The Free Software Bazaar

Interview with Axel Boldt

Axel Boldt's Free Software Bazaar was the first realization of these ideas. It opened in the fall of 1998; within six months, it collected over \$25,000 US in offers and over \$1200 in payments towards Open Source projects. The site works by letting users browse a list of existing offers. If a user is interested in sponsoring an existing project or creating a new one, they send e-mail to Axel. He then adds their offer to the listing page.

These offers can be claimed by the first developer to successfully complete the work. Axel then notifies the sponsors, asking them to send payment directly to the developer.

CoSource.com

The Free Software Bazaar is a great service to the Open Source community. However, a huge ongoing effort is required to maintain momentum and grow the movement into something powerful.

For cooperative funding to become a significant force in the Open Source market, the achievements of the Free Software Bazaar must be multiplied many times. Managing the demands of so many parties is a difficult problem. A cooperative funding service needs to be innovative in solving the confidence and communication problems between sponsors and developers. It should be convenient and simple. It must be professional and build a strong record of trust. In the end, it is essential to attract and maintain a critical mass of sponsors and developers.

CoSource.com is an attempt to create a service that meets these demands. It is a commercial enterprise created to provide the range of services required to make cooperative funding a success for buyers, developers and the Open Source community in general.

It intends to:

- Publicize to better achieve a critical mass of sponsors and developers for each project.
- Have staff work with corporations directly to encourage sponsorship and development of open source.
- Provide VISA/MC/AMEX credit card processing to make payment convenient (especially for non-US sponsors).
- Automate as much as possible with HTML forms and a database back-end.
- Provide a stable web address and organization for long-running projects.
- Foster trust between buyers and developers through simple, standard and legally binding on-line agreements.
- Provide cash advances to developers with a proven track record when they begin work.
- Provide web space to trumpet the financial contributions of corporations and individuals towards particular projects.
- Prevent duplication of effort by having sponsors collectively hire a single developer to work on a project.
- Allow sponsors to back out any time until a developer is finally selected.
- Allow full control for sponsors to select the developer, development schedule, source code license, etc. that suits them.

If all these goals can be achieved, cooperative funding will provide effective answers to the questions, "How does one make a living on free software?" and "Who is motivated to innovate?"

The answers are that innovation is funded directly by users who pay for new features, which in turn supports a small army of independent software developers.

How Does It Work for Sponsors and Developers?

It starts with an unfulfilled need. Maybe it's a driver for a USB scanner, a plug-in to convert Excel spreadsheets or a port of some game to Linux. The user goes to <http://www.cosource.com/> and finds the project to develop this feature. If it is not there, they can add it with a form.

What is it worth to them for someone to develop this software? Whether it is \$10 or \$1000, the customer sponsors the project for that amount. This is not something done lightly. A buyer is making a firm commitment to pay up if the software is developed.

Other motivated sponsors come along and do the same. CoSource goes out to corporations and seeks to supplement individual sponsorships with a few large ones. Let's say the project is an HP scanner driver. While HP isn't yet ready to pay the full cost of developing a Linux driver, they may be willing to pay for 50% or 25% of the effort.

Developers, meanwhile, browse these same lists to identify projects in their area of expertise. Suppose a developer has done a converter for the Excel file format before. That developer fills in a form to bid on the project, answering the following questions:

- How much would I have to be paid to do this work?
- What is my most conservative estimate for time to completion? What license would it be released under (e.g., GPL, BSD, etc.)?
- Who will judge the final product for completeness and quality (e.g., a known and trusted third-party authority)?
- What will be the URL of the project's web page?

CoSource then prices the bid for display on the system, marking up the bid for transaction costs, historical sponsor fraud, advance payments, project risk, etc.

As bids are entered, sponsors are notified to evaluate them. They submit a simple yes/no form in response. Voting yes to one or more bids is a final commitment involving a legal agreement to follow through if this developer succeeds. The first bid that gains sufficient sponsorship wins. From that time forward, sponsors are not permitted to back out and shortchange the developer.

The winning developer then begins work on the project, providing updates on their project web page.

At some point, the developer believes he is complete. A release is done and judged to determine if it matches the requirements stated in the original project description. If the release fails, the developer may try again many times until their committed schedule runs out. If that unfortunate event happens, the project goes back to the sponsor/bid phase.

If the release passes, the project is complete! Sponsors are notified to fulfill their commitments. They can easily pay by credit card. Finally, payments are consolidated and a single check is mailed to the developer.

What Are the Goals?

Obviously, this process is more complicated than a typical software-buying experience. In return, the consumer gains much more control over the quality and time frame of work. If you needed one of the new features Windows 2000 provides, you would have to wait two to three years after the initial promised ship date to get them. How can a corporation plan ahead for software rollouts with such uncertainty?

Cosourcing puts more control over features, schedule and quality in the hands of the consumers.

Obviously, this system is not intended for charity or non-profit activity. Rather, it is intended to be the most effective way to outsource development of software and share that cost with other motivated buyers. It is intended to be a way for non-developers to "scratch their own itch". It is intended to be a fertile breeding ground for hundreds of individual and corporate developers. It is intended to make the funding of Open Source a collaborative effort in the same spirit the development process is in today.

In general, it is intended to empower end users to spend their hard-earned money making free software do what they need.

Making a New Market

On one end of the software market spectrum is "closed" software, where intellectual property is licensed on a per-copy basis. On the other end is free software, where intellectual property is created without payment and voluntarily given to the community at large.

Both of these will continue to grow and thrive. On one hand, closed software will continue to be a billion-dollar market. On the other hand, innovative free software will continue to be developed by students, hobbyists and professionals for various reasons. Both systems make sense and they will continue to compete with each other. But there is a possibility for a vibrant third market. One which blends and bridges the differences between the other two. One that brings the free market to free software.

This market will sell software as a service rather than a product, so it will be compatible with Richard Stallman's original and ongoing vision for public license code. This market will serve the needs of end users by driving

innovation in the areas that matter most to them. This market will bring financial vitality to free software, so thousands of individuals and small companies can make their living developing it.

Software in 2010—A Look Into the Future

This vision may turn out to be a pipe dream. Consumer psychology has rarely dealt with a case in which a group of consumers pay for the development of a product, then allow that product to be given away freely from that time forward. Psychologically, this is a strange mix of self-interest and altruism.

CoSource.com and others are going to put it to the test. If you've ever complained about some missing piece of free software, now you can put your money where your mouth is. Will you?

Resources



Bernie Thompson is one of the founders of CoSource.com. He lives just down the hill from Microsoft and believes that a great and healthy rivalry has begun where the big winner will be end users. Send comments and ideas to bernie@cosource.com.

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

[Advanced search](#)

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

[Advanced search](#)

Interview: Hans L. Knobloch of IGEL

Marjorie Richardson

Issue #64, August 1999

A talk with the head of the company that invented the thin client.



Hans L. Knobloch is President and CEO of IGEL LLC (<http://www.igelusa.com/>) and owns a 10 percent interest in the company. He was born in Germany and moved to the U.S. in 1993 to head up IGEL, founded in 1992. He holds an engineering degree in Computer Science and a degree in Physics and Electronics. Before moving to the U.S., he was VP of Germany's second largest PC network and UNIX support company. He also owned a consulting company which specialized in small business computing. I interviewed him via e-mail in mid-May.

Margie: When did you start up IGEL? Tell us a bit of the company's history.

Hans: The company was founded in October of 1992 by the same partners who owned and still partially own IGEL GmbH (<http://www.igel.com/>) in Germany. IGEL started out as a distributor for a SCO UNIX Multi-Console Solution. In 1992, IGEL realized the limitations of this technology and developed the first character-based Ethernet TCP/IP terminal. The product, Etherminal 2C, was released in 1993. The successors of Etherminal 2C, Etherminal C and N are still selling today, creating a substantial part of IGEL's revenue. In 1993 IGEL

developed the first (Flash-) Linux-based workstation, Etherterminal 3X. IGEL actually invented the thin-client concept. This product had the same concept and basic functionality of our current Etherterminal thin-client product line. In 1996, IGEL developed the Linux boot driver to boot Linux from an M-System's DiskOnChip (DOC) Flash Memory. The driver has been released under the GPL to the Linux community and can be downloaded from M-System's server.

IGEL developed a compression technology to compress a complete Linux system, including Netscape Communicator, into as little flash memory as possible. This resulted in IGEL's first commercially available thin client, which was released in 1998. In mid-1998, IGEL added support for the Citrix Metaframe. RDP support for the Microsoft Terminal Server will be available sometime in July.

The next and final step was the development of a read-only flash Linux file system. The development of these core technologies resulted in a new Linux-based embedded OS which is named IGEL JNT. IGEL JNT, besides being the OS for IGEL's thin clients, is available to OEMs (original equipment manufacturers) for licensing. JNT is used for TV Internet set-top boxes, OEM thin clients, network appliances, Linux/UNIX workstations and embedded computer systems for industrial and residential applications.

In 1999, IGEL released a JNT system builder kit which again can be licensed by OEMs, allowing them to develop their own embedded Linux devices or modify our thin clients to accommodate the OEM's proprietary application. IGEL, of course, does not charge any licensing fees for the actual Linux part, only for the compression technology, the read-only file system and the system builder kit.

Currently established OEM partnerships include Siemens, Schneider Cybermind, VxL and Mercedes Chrysler. With over 240,000 JNT units sold within the last 12 months, IGEL is becoming the leader in thin-client computing.

In 1998, IGEL GmbH Germany sold 51 percent of its company to Germany's largest Internet technology company, Infomatec. IGEL GmbH also became a 90 percent owner of IGEL LLC (USA), while 10 percent of IGEL LLC's ownership has been transferred to me. The consolidation of Infomatec and IGEL provides a solid base for the new partnership to push IGEL's JNT even further, setting a standard for non-Windows-CE-based thin clients. The new partnership also lays the groundwork to take IGEL public in 2001.

Margie: Why did you choose Linux as your installed operating system?

Hans: Linux is one of the most flexible open-source operating systems on the market. It is incredibly stable and performant (no matter what certain

benchmarks insinuate). It is the most flexible OS on the market, and if your readers follow IGEL's developments, they will soon be able to enjoy incredible new JNT-based products.

Margie: The Etherminal looks just as “sexy” as the Netwinder. In what way is it different? How is it better?

Hans: IGEL's Etherminal series is a completely different “animal” than the Netwinder. It targets a different market and comes with a much more powerful set of functions and connectivity options. Etherminals do not have or need any hard drive or other mechanical storage device, and are based on legacy PC-compatible technology. The flexible hardware and software concept allows IGEL to react to the increasing demand for processing power in workstations and thin clients within a very short time, giving IGEL a tremendous advantage in time to market. The read-only file system protects the systems resources, making Etherminal thin clients the most safe and secure desktop clients for server-centric computing on the market. Etherminals and their JNT OS are an ideal base for OEMs to start development for almost any kind of dedicated embedded Linux network device. We at IGEL do not see competitors as our enemies, but as an opportunity to license our Etherminal and JNT technology to them.

Margie: Tell us about your product line; how these machines can be used, and who your potential customers are.

Hans: As of today, IGEL offers three thin-client devices. Two of them are based on the same hardware, while the third one is based on more up-scale hardware.

Etherminal W is a JNT-based thin client to be used as a Citrix Metaframe ICA client, UNIX workstation or a character-based network terminal. It supports XDMCP, DHCP, ICA 3 and remote command **rsh**. It is completely self-booting, and therefore does not require a bootp server. It is centrally managed and configured. The hardware is based on a Media GXi 180MHz processor and is equipped with 16MB of RAM, 8MB DOC flash memory, sound and 4MB of video memory.

Etherminal J comes with the same set of functionality, but adds an embedded Netscape Communicator 4.5 with all the bells and whistles, including Netscape's JVM. The local Netscape is ideal for saving network bandwidth and eliminates the restrictions of 256 colors in an ICA session while browsing the Internet. It also allows Java applications to be run locally. On the hardware side, the unit comes with 32MB of RAM and 16MB of DOC flash memory.

Etherminal J+ comes with the same feature set as Etherminal J, also including the local Netscape Communicator 4.5 and JVM. The hardware is based on a socket 7 design to accommodate processors from 100MHz (Intel Pentium is default) to 300MHz (AMD K6). RAM size is 64MB and DOC flash memory is 16MB.

While Etherminal W and J come with an integrated power supply, Etherminal J+ has an external power supply which eliminates the need for a power supply fan to cool the device. The form factor benefits from this design as well—the unit takes up less than 50 percent of the real estate than that of an Etherminal W or J.

IGEL is currently porting Windows NT Terminal Server's RDP to Linux. Once available, it will be integrated as an additional client protocol into all of IGEL's thin clients. The RDP client will then be distributed under the GPL. IGEL is also a partner in Citrix.

Typical end-user applications are POS, Citrix Metaframe and Winframe ICA terminal, WTS terminals, web surfing, UNIX workstations and Linux workstations. Since IGEL is selling through the traditional computer reseller channel, any distributor, reseller, VAR, ISP or ASP is a potential customer. Of course, we have special conditions for government and educational institutions.

Margie: Thin clients certainly seem to be popular at the moment. Has business been increasing along with the rise in Linux popularity?

Hans: I believe it is actually the other way around. Thin client deployments seem to benefit Linux as a desktop OS and hopefully in the future as a enterprise server as well. Of course, the increasing popularity of Linux has helped us to justify using it as our development platform.

Margie: Do you expect the upward trend to continue? For how long?

Hans: I have been a firm believer for many years that the server-centric computing model is the best way to get the masses to use a computer. It is certainly the most economical way to distribute computing power and applications to the end user.

The client/server model never fulfilled its promises when it comes to productivity issues, low TCO and more effective work environments. Client/server deployments in big corporations create support and help desk nightmares and lead to productivity problems. It even produced legal nightmares due to end users infiltrating corporate networks with unauthorized

software. PCs on every user desk, regardless of his or her job function, is in many cases a very costly overkill. It creates huge problems with security and virus issues if end users have too much freedom in what they use or bring into corporate networks.

Today's thin-client technology marries the advantages of server-centric computing and PC computing without the disadvantages of the old mainframe technology. It opens up new opportunities for ASPs and ISPs alike to work together to get the applications to the end user. Linux can play a very vital role in this new computing model, but it desperately needs productivity applications to put a real dent into the current Windows-ruled desktop world. I believe thin-client computing in whatever shape or form will stay around for a very long time to come, and I hope Linux will play an increasing role in this server-centric computing model.

Margie: Any parting thoughts?

Hans: Linux and its open-source movement is the right way to go, and I foresee a very bright future for this operating system. Today's computing world desperately needs innovation—something we don't get any more from current software and hardware vendors. How else can it be explained that we still have to live with an over 20-year-old outdated PC hardware concept and outdated bloated software technology? I personally believe Linux has the potential to become a major player and make big changes in the way we will be computing in the future. What Linux has achieved within a few years is a hint as to what is possible in the future. *Linux Journal* certainly plays an important role in the development of this new opportunity.

We at IGEL appreciate the continuous support of your publication. *Linux Journal* plays an important role in our efforts to communicate our message to both the Linux and thin-client worlds.

Margie: Thanks very much for talking to us.

[Acronyms](#)

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

[Advanced search](#)

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

[Advanced search](#)

AbiWord: AbiSource's Open Source Word Processor

Craig Knudsen

Issue #64, August 1999

A cross-platform commercial application is giving away their source—here's the story.

There is a growing number of quality open-source efforts for both word processors and office suites. Both GNOME and KDE are actively developing a variety of office applications. Although these efforts may end up with commercial-quality applications (and some of the developers are being paid to work full-time on them), they are not being developed by a for-profit company.

AbiSource is currently developing the open-source word processor, AbiWord. AbiSource is indeed a business, rather than a network of volunteer open-source developers: their goal is to be profitable using paid employees as the core team of developers.

Although the other open-source efforts are a great asset to Linux in general (as are commercial companies distributing free binaries), a company releasing source code for their only application deserves a closer look.

AbiSource

AbiSource began in January of 1997 as a contracting firm with no plans for creating products. The company decided to shift its focus toward product development, but it was not until early 1998 when the Open Source movement gained momentum that they developed a new business plan. By the end of 1998, they had already released the first developer version of AbiWord under the GNU General Public License (GPL).

AbiSource's mission is "to become the leading provider of open-source desktop productivity applications." Their plan is to develop a cross-platform, open-source office suite called AbiSuite. Their first application under development is the word processor AbiWord.

AbiSource is betting 100% of their future on Open Source. One of the difficulties in open-source development is getting volunteer developers involved. (Netscape's Mozilla browser development is still almost entirely done by Netscape/AOL engineers.) Asked how to rate the success of AbiSource's open-source development model, AbiSource founder Eric W. Sink replied,

People tend to look at projects like Apache and Linux with the assumption that community development is a panacea. When you sit down to duplicate their experiences in the application space, the first thing you do is resolve to eliminate all obstacles which would prevent widespread community participation in the development. After that, you realize just how successful Apache and Linux are. It's not enough to *let* the community participate. You've got to create something cool enough that others find it attractive. We're just now getting to the point where our project has serious participants beyond the original core group.

Open-source development has already helped the progress of AbiWord. A volunteer developer provided a BeOS port. Others provided translations to allow AbiWord to be localized for four languages other than English.

AbiWord

Figure 1. AbiWord Screenshot

What separates AbiWord (see Figure 1) from the other open-source word processors available for Linux? It is also available for Windows and is a WYSIWYG (what you see is what you get) editor.

Open Source software for Windows is uncommon. Sure, you can download plenty of free software for Windows, but source code for Windows applications is the exception rather than the rule. I suppose we can thank Microsoft for not bundling a free compiler with each version of Windows. What's an end user going to do with source code and no compiler? Even AbiWord requires Microsoft Visual C++ (\$109 and up) to build.

AbiWord's source code is approximately 90% cross-platform and around 10% platform-specific. After evaluating the wxWindows cross-platform GUI toolkit (and nearly choosing it), they decided to implement their own cross-platform library. AbiWord is written in C++ and uses the GTK toolkit for its UNIX GUI (and the Win32 API for its Windows GUI). Ports are underway for both BeOS and Macintosh. In fact, you can see the directories "beos" and "mac" in the latest source code tree.

Windows users expect word processors to be WYSIWYG. This allows the user to see what the printed document will look like, as they edit it. In the UNIX world, this is more difficult to accomplish, and therefore less common. In order to achieve WYSIWYG under Linux, you need to use Type 1 PostScript fonts for both display and printing. (PostScript is a page description programming language for printers.) Many X servers (such as XFree86 for Linux) include PostScript fonts, but do not include the associated metric files needed to use the fonts for printing. AbiWord users are required to install a set of fonts taken from GhostScript (an Open Source PostScript interpreter) and update their X server configuration to recognize the new fonts. The PostScript fonts are used by the X server to display the text and by AbiWord to generate PostScript output to the printer. As of AbiWord 0.5.5, installing the PostScript fonts involves hand editing files, but this process will be automated in future releases.

The Business Model

Unlike Netscape, AbiSource does not have millions of dollars in other product sales to fund development. Until AbiWord 1.0 is ready, AbiSource will have to rely on a combination of revenue from their previous contracting work and funding from outside sources. AbiSource currently has some investors and is in the process of looking for more.

When AbiWord 1.0 ships to end users later this year, AbiSource will begin offering services and support to generate revenue. There is a large and always-growing demand for desktop office tools. Eric believes AbiWord will attract users "who value the advantages offered by community-developed Open Source software."

Future Development

Although I found AbiWord 0.5.5 stable enough to write this article, it is still considered a developer release. AbiWord 1.0 should be available by the end of the year. AbiSource plans to continue development of its AbiSuite software. The next application will be AbiFile, a personal database manager and then a presentation manager called AbiShow. Gnumeric (the spreadsheet under development for the GNOME project) will serve as AbiCalc. AbiSource's role in Gnumeric development will be to make it cross-platform. Additionally, support for importing and exporting documents in formats like RTF, MS Word, Word Perfect and HTML will be a high priority. These tools will be developed as modules and will be ideal opportunities for outside developers to contribute to the project.

Asked where he sees the company in a few years, Eric replied,

We expect to succeed. The existence of a full-featured, cross-platform, open-source office suite will bring substantial change to the software industry. We expect to be the company that brings that change. It's going to be a fun ride.

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. Craig has been using Linux for both work and play for three years. 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](#)

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

Dynamic Load-Balancing DNS: **dlbDNS**

Harish V.C.

Brad Owens

Issue #64, August 1999

An attempt to solve the problem of network traffic congestion by adding a dynamic load-balancing feature to the existing DNS.

The rapid growth of computer literacy has led to a dramatic rise in the number of people using computers today. This rise has resulted in the development of intense computation-oriented and resource-sharing applications. These factors together play a prominent role in increasing the load across the Internet, causing severe network traffic congestion. This phenomenon, though dynamic in nature, causes a lot of user frustration in the form of slow response times and repeated crashing of applications.

Developing servers with more capacity and capability of handling this traffic is one way to solve the problem; another is to distribute client requests across multiple servers. This second method is an elegant way of handling this problem, since it uses existing resources and avoids scenarios in which some servers are overloaded while the rest of them are idle. The need for distributing requests across servers is further strengthened, considering:

- Each TCP session eats up 32 bytes of memory (a general rule of thumb), causing a server that has 32MB of RAM to theoretically support one million simultaneous connections (see Resources 2).
- Given a number of servers, users always log in to their favorite server while overlooking the load on that server.

Distributing a request across servers can be implemented by monitoring the servers regularly and directing the request dynamically to the *best server*. This way of dynamically directing a request across multiple servers based on the server load is called dynamic load balancing. This feature can be added to the pre-existing Domain Name Service (DNS), as it already plays a prominent role in

resolving client requests and can be configured to direct client requests across multiple servers in an effort to avoid network traffic congestion. Here, *best server* refers to the server with the best rating based on a rating algorithm to be explained later.

Snapshots

We will explain the design, implementation and benefits of a dynamic load-balancing DNS, dlbdns, which extends DNS.

Minimum Requirements for dlbdns

Load-Balancing Models

Four load-balancing models are available. First, RFC 1794 (see Resources 1) describes a load-balancing method using a special zone transfer agent that obtains its information from external sources. The new zone then gets loaded by the name server. One problem with this method is that between zone transfers, the weighted information is essentially static or possibly handed out in a round-robin fashion. This method also doesn't allow a virtual/dynamic domain where a response is created dynamically based on the name being queried (see Resources 4).

The second model is a dedicated load-balancing server which intercepts incoming requests and directs them to the best server. This design employs virtual IP addresses for internal use by the load-balancing server. One problem with this is it adds another server to the existing cluster of servers to be monitored, instead of utilizing the available resources.

A third model is a remote monitoring system that monitors the performance of different servers and provides feedback to the DNS. This design helps detect problems not visible internally, and provides truer access time measurements and easy detection of configuration errors that affect external users. The major problem here is the dependency on the remote network to monitor and deliver data (see Resources 5).

Last is an internal monitoring system that monitors the performance of the servers and provides feedback to the DNS. Its major advantages are easy maintainability and administration, closeness to the source of addressable problems and no security hazards (see Resources 5). This design is implemented in dlbdns.

Load-Balancing Algorithms

Initially, load-balancing was intended to permit DNS agents to support the concept of machine clusters (derived from the VMS usage) where all machines were functionally similar or the same. It didn't particularly matter which machine was picked, as long as the processing load was reasonably well-distributed across a series of actual different hosts. With servers of different configurations and capacities, there is a need for more sophisticated algorithms (see Resources 1).

“Round-robin algorithm A” can distribute requests in a round-robin fashion evenly across servers. Although the requests are handled dynamically, the problem is the total ignorance of various performance characteristics.

“Load-average algorithm A” can distribute requests across servers based on the server load. This design is very simple and fairly inexpensive, but fails miserably if servers vary in configuration and potential.

“Rating algorithm A” is based on the number of users and load-average shown below. This algorithm is reasonable, as its rating favors hosts with the smallest number of unique logins and lower load averages (see Resources 4). This rating algorithm is implemented in dlbdns to determine the *best server*.

```
WT_PER_USER      = 100
USER_PER_LOAD_UNIT = 3
FUDGE            = (TOT_USER - UNIQ_USER) * (WT_PER_USER/5)
WEIGHT          = (UNIQ_USER * WT_PER_USER) + (USER_PER_LOAD_UNIT * LOAD) + FUDGE
```

where the variables are

- **TOT_USER**: total number of users logged in
- **UNIQ_USERS**: unique number of users logged in
- **LOAD**: load average over the last minute, multiplied by 100
- **WT_PER_USER**: pseudo-weight per user
- **FUDGE**: fudge factor for users logged in more than once
- **WEIGHT**: rating of the server

dlbdns Implementation

To get started, we downloaded BIND 8.1.2 from the Internet Software Consortium (www.isc.org/bind.html). Initially, time was spent installing and understanding DNS. DNS was installed on odie.cs.twsu.edu, a stand-alone Linux workstation.

Listing 1. named.hosts.wsu

During configuration, a new attribute called DNAME was added to distinguish the hosts taking part in dynamic load-balancing. Listing 1 is a snapshot from named.hosts.wsu, containing information on all hosts in a particular zone. In this listing, the set of hosts kira.cs.twsu.edu, sisko.cs.twsu.edu and q.cs.twsu.edu take part in dynamic load-balancing for <http://www1.cs.twsu.edu/>. The set of hosts kira.cs.twsu.edu, mccoys.cs.twsu.edu and emcity.cs.twsu.edu take part in dynamic load-balancing for <http://www2.cs.twsu.edu/>. The set of hosts kira.cs.twsu.edu, sisko.cs.twsu.edu and deanna.cs.twsu.edu take part in dynamic load-balancing for <http://www3.cs.twsu.edu/>. Hosts kira.cs.twsu.edu and sisko.cs.twsu.edu belong to multiple groups.

Server-Side Algorithm

Here is the algorithm we added to the pre-existing DNS feature. If the service requested is of type DNAME, do the following:

1. Determine the set of participating servers for this service.
2. Request ratings from all participating servers by establishing a concurrent connectionless (UDP) connection with each server.
3. Using the ratings returned, determine the best server.
4. Handle error conditions such as "server is too busy to return the rating within the time frame", "the rating returned by the server gets lost on its way back to the dlbDNS", "all servers have same rating" and "a server is down".

Rating Demon Algorithm

A rating daemon runs on each server taking part in dynamic load balancing. Here is the algorithm:

1. Receive request for rating from dlbDNS and respond by returning the host rating.
2. Calculate the host rating once every minute rather than calculating it at the time of request, as quick response time is a most important feature.
3. Ensure the host rating is updated every minute, independent of the dlbDNS request.
4. Handle error conditions such as dlbDNS closing the UDP sockets without waiting for host response.

Figure 1. dlbDNS.gif

Model

Figure 1 shows the functionality of dlbdns. The path traced by C indicates the process of updating the server rating by the rating daemons. The path traced by B indicates the communication between dlbdns and the rating daemons to determine the best server. The path traced by A indicates the path traced by the user request. HOST 1 has a better rating than the other two hosts, so the user request gets directed to HOST 1.

dlbdns Benefits

Implementing dlbdns provides efficient utilization of system resources and ensures that facilities newly added to the existing network will be utilized. Since DNS is used, applications such as FTP and TELNET will also utilize dlbdns.

dlbdns Current Implementation

Uneven distribution of load across servers has been a major problem in the Computer Science department of Wichita State University. bugs.cs.twsu.edu, kira.cs.twsu.edu, roger.cs.twsu.edu and sisko.cs.twsu.edu are four Linux servers available for students in the department. These servers vary in potential and configuration.

dlbdns was installed in December 1998 to effectively utilize the servers. lion.cs.twsu.edu, the actual DNS server, was made to direct DNAME requests toward odie.cs.twsu.edu where dlbdns was installed. The lines added to the configuration file were:

```
;
bestlinux      IN      DNAME  bugs.cs.twsu.edu.
bestlinux      IN      DNAME  kira.cs.twsu.edu.
bestlinux      IN      DNAME  roger.cs.twsu.edu.
bestlinux      IN      DNAME  sisko.cs.twsu.edu.
;
```

Here, the **bestlinux** attribute was added to handle non-web requests from applications such as TELNET and FTP.

Future Work

Currently, the **gethostbyname** system call fails within the BIND code. This problem is avoided by using a configuration file with a list of host and IP addresses. We'd like to find a better solution.

The rating algorithm is still not complete. An algorithm that takes into account the number of processors, CPU and memory utilization would make the rating algorithm more efficient.

At this time, only Linux servers can take part in the dynamic load-balancing scheme, as the rating algorithm uses files in the /proc file structure. A more extensible design is needed.

Resources

Acknowledgement



Harish V.C. (harish@acm.org) is a graduate student in the Computer Science department of Wichita State University. His research interests include computer and Internet security, networking and operating systems. He is currently working as an intern at IBM.



Brad J. Owens (bjowens@cs.twsu.edu) is a faculty member in the Computer Science department at Wichita State University. His research interests include computer and Internet security, high-speed networking, parallel and distributed programming.

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

[Advanced search](#)

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

[Advanced search](#)

Advanced “New” Labels

Reuven M. Lerner

Issue #64, August 1999

Improving the way your site handles “new” labels using the popular Apache modules `mod_perl`.

Last month, we looked at those pesky “new” labels that webmasters like to put on their sites. The intention is good, pointing us to documents we are unlikely to have seen before. In practice, the “new” labels are artificial, telling us when the document was last published, rather than whether it is actually new to us.

The techniques we explored last month—server-side includes, CGI programs, and templates—were interesting, but inefficient and slow. This month, we will look at ways to speed up our performance by using `mod_perl`, the Perl module for the Apache server.

What Is `mod_perl`?

We have discussed `mod_perl` on previous occasions in this column, but it is worth giving a quick introduction for those of you who may have missed it. The Apache server is built of modules, each of which handles a different part of the software's functionality. One of the advantages of this architecture is that it allows webmasters to customize their copy of Apache, including or excluding modules as necessary. It also means programmers can add functionality to Apache by writing new modules.

One of the most popular modules is `mod_perl`, which puts a copy of the Perl language inside the Apache server. This provides functionality on a number of levels, including the ability to set the configuration directives in Perl (or conditional, depending on whether or not certain Perl code executes). More significantly, it allows us to write Perl modules that can modify Apache's behavior.

When I say “behavior”, I mean both the behavior users see, displaying documents and responding to HTTP requests, and that which takes place behind the scenes, ranging from the way authentication takes place to the way logging is done.

Each invocation of a CGI program requires a new process, as well as start-up time. By contrast, `mod_perl` turns your code into a subroutine of the Apache executable. Your code is loaded and compiled once, then saved for future invocations.

When we first think about what happens to an HTTP request when it is submitted to Apache, it seems relatively simple. The request is read by Apache, passed to the correct module or subroutine and returned to the user's browser in an HTTP response. In fact, each request must travel through many (over a dozen) different “handlers” before a response is generated and sent. **`mod_perl`** allows us to modify and enhance any or all of these handlers by attaching a Perl module to it. The handler most often modified is called `PerlHandler`. Other more specific handlers are given other names, such as `PerlTransHandler` (for URL-to-filename translation) and `PerlLogHandler` (for log files).

This month, we will look at a number of `PerlHandlers` that will make it possible to create truly useful “new” labels for our web sites.

A Simple Version

The first `PerlHandler` we will define is rather simple: it puts a “new” label next to any link on a page. This is not a particularly difficult task or a good use of `mod_perl`. However, it does gently ease us into writing a Perl module for `mod_perl`, and it will form the basis for future versions we will write.

Our module begins much the same as any other module, declaring its own name space (`Apache::TagNew`, in this case), then importing several symbolic constants from the `Apache::Constants` package. The module defines a single subroutine, called “handler”. This is the conventional way to define a handler under `mod_perl`; that is, create a module with a “handler” subroutine, then tell Apache to use that module as a handler for a particular directory.

We instruct Apache to invoke our handler in the configuration file `httpd.conf`. For example, my copy of `httpd.conf` says the following:

```
PerlModule Apache::TagNew
<Directory /usr/local/apache/share/htdocs/tag>
  SetHandler perl-script
  PerlHandler Apache::TagNew
</Directory>
```

The PerlModule directive tells Apache to load the Apache::TagNew module. The <Directory> section tells Apache that the /tag subdirectory of my HTML content tree should be treated specially, using the **handler** method of Apache::New instead of the default content handler. Once we activate our module by restarting our server (or by sending it a HUP signal), any file in the /tag directory will be handled by Apache::TagNew, rather than Apache's default handler.

The first thing handler must do is retrieve the Apache request object, traditionally called **\$r**. This object is the key to everything in mod_perl, since it allows us to retrieve information about the HTTP request, the environment, and the server on which the program is running. We also use **\$r** to send data back to the user's browser.

Our method is expected to return one of the symbols we imported from Apache::Constants. Returning **OK** means we successfully handled the query, data has been returned to the user's browser, and Apache should move to its next stage of handling the request. If we return **DECLINED**, Apache assumes our module did not handle the request and it should find some other handler willing to do the job. There are a variety of other symbols we can return, including **NOT_FOUND**, which indicates that the file was not found on our server.

Listing 1.

In Apache::TagNew (see Listing 1), we normally return **OK**. We return **NOT_FOUND** if an error occurs when opening the file, and **DECLINED** if the file does not have a MIME type of "text/html". Hyperlinks are going to appear only in HTML-formatted text files, so we can save everyone a bit of time and energy by letting another handler take care of other file types.

The rest of the handler works by reading the contents of the file, then replacing them with our new and improved version. We append a "new" label after every , which comes after each hyperlink. In this way, every hyperlink is tagged as new.

Printing "New" on New Files

Of course, the point of this project is not to print "new" next to all links, but rather next to new ones. In order to do that, we will need to look at each link in sequence and check to see if it is on our system. If it is, we will check when the associated file was last changed. If that file was changed within the last week, we will tag it as new; otherwise, we will leave it alone.

Listing 2.

In order to do this, we will write another subroutine, which takes care of identifying a link and adds the appropriate text when necessary. That is, the subroutine will take a URL as input and will output either the same URL or the URL with a “new” label appended. Listing 2 is our new version of `Apache::TagNew` and it contains just such a subroutine, called `label_url`. The `label_url` subroutine expects to be invoked with three arguments: `$r`, the Apache request object, `$url`, the URL of the hyperlink in question, and `$text`, the text that goes between the `<a>` and `` tags of the hyperlink.

We can know whether a file has changed only if it is on our system. Rather than parse through the URL, I decided to take the simple way by checking whether the URL in question begins with “`http://`”. If it does, then we assume the URL points to a file on a different system, and we ignore it, returning the URL and text in their original states.

If the URL begins with any other characters, it is assumed to point to a file on our system. We use `$r` to retrieve the value of the document root directory, namely the directory under which all URLs are stored. This module will work regardless of whether your web documents are under `/usr/local/apache/share/htdocs`, `/etc/httpd/htdocs` or even `/usr/local/bin`. `$r` retrieves the information from the `httpd.conf` file, which also means the module does not need updating if you decide to move the document root.

We then check to see whether the file was modified within the last seven days, using Perl's `-M` operator to get the last modification time. Luckily for our purposes, `-M` returns its result in days rather than seconds; so, we can simply compare the returned result with 7 and add the label as necessary. If the file was unmodified in the last seven days, the `$label` variable remains undefined and turns into the empty string later.

Our program returns the modified URL, much as it did in the previous version of `Apache::TagNew`.

We can evaluate this subroutine over every hyperlink in a document with `s///`, Perl's substitution operator. We give `s///` three modifiers: `g` performs the operation globally, `i` ignores case and `e` replaces the initial text with the result of evaluating the substitution:

```
$contents =~  
s|<a\s+href=[\'"]?(\\S+?)[\'"]?\\s*>(\\s\\S)+</a>  
|label_url($r, $1, $2)|eig;
```

The above regular expression is difficult to understand, so let us examine what it does in greater detail. We make the regexp more readable with the “`x`” modifier, which allows us to insert whitespace inside of it. We look for the opening `<a>` and closing `` tags and extract from them the URL, which is

grouped inside the first set of parentheses, and the link text, which is grouped inside the second set of parentheses. We use Perl's non-greedy operators to ensure we get only the necessary text. Otherwise, such things as quotation marks might be included in our link text.

We then invoke the subroutine `label_url`. We pass it three arguments: `$r` (the Apache request object), `$1` (the URL we grabbed from the first set of parentheses) and `$2` (the link text we grabbed from the second set of parentheses). Whatever `label_url` returns is substituted for the text we originally found. In this way, we can optionally insert a label into the text of the document.

Storing Information Across Sessions

The above system has several advantages, but it fails to keep track of when users went to a particular link. In other words, it is terrific at keeping track of a countdown timer for a particular URL, tagging it as new for the first seven days. But once again, we want to produce a “new” label when the document is new to a particular user. What if I have not visited a site in three months? Then all of the content is likely to be new, and “new” labels will be on everything. By contrast, if I visited the site two hours ago, only those labels that have changed since my visit will look different.

Keeping track of such information would require us to keep state across HTTP requests, so that we could keep track of which links were seen by a particular user. Unfortunately, HTTP is a stateless protocol, which means we cannot save such information. HTTP requests and responses take place in a vacuum, neither storing information for the next transaction nor retrieving information from a previous transaction.

HTTP's stateless nature has created problems for web programmers and designers who wish to create useful applications and has led to a number of clever solutions. Perhaps the most famous solution is the use of HTTP cookies, which allow a web server to store information on the user's computer. Each time the user submits an HTTP request to that server, all cookies previously stored are sent along with the request.

Cookies can store information in several ways. One is by putting the information inside the cookie, thus giving the server immediate access to further details about the user as part of the request. But this quickly becomes cumbersome if you have too much data. For this reason, it is common to use a table in a relational database to keep track of user information. If we define a primary key (i.e., a column guaranteed to be unique) for that table, we can store as much information as we like in the table.

Accessing a table in this way can be cumbersome, since it involves many database storage and retrieval operations. Luckily, we can use the `Apache::Session` module to handle such things. `Apache::Session` works with `mod_perl` programs to store and retrieve information across HTTP transactions.

We can retrieve the cookie in our handler using the `header_in` method. Notice how we are working with the raw cookie, meaning we must use `s///` to retrieve the value of interest:

```
my $id = $r->header_in('Cookie');
$id =~ s/SESSION_ID=(\w*)/$1/;
```

Once we have done this, we can use `Apache::Session::DBI`, the module that connects sessions to a database table. We use Perl's `tie` routine, which creates a connection between a variable and a module, to provide a seamless connection:

```
tie %session, 'Apache::Session::DBI', $id,
    {
        DataSource => 'dbi:mysql:test',
        UserName   => 'username',
        Password   => 'password'
    };
```

You might recognize the three attributes in the above code fragment from DBI, the Perl database interface. DBI works with many different relational databases, thanks to its use of database drivers for specific databases. The above example uses the MySQL database, which I use for many of my database tasks. This example uses the “test” database to store our session information. While “test” is a good place for demonstration databases, you would be wise to put production databases somewhere else.

`Apache::Session` cannot create a table in MySQL for you. Before using the above code, you will need to create a table in which `Apache::Session` can store its session information. Here is the recommended table definition, from the `Apache::Session::DBI` manual pages:

```
CREATE TABLE sessions (
  id char(16),
  length int(11),
  a_session text
);
```

Using `Apache::Session`

Once our handler has retrieved the user's ID from a cookie and established a connection with the database, we can store and retrieve session information at our convenience.

We can store information about this user in **%session**, the hash to which we tied `Apache::Session`. Each time our handler is invoked, we can retrieve information about this user based on his or her ID. For example, we can store a value with:

```
$session{"foo"} = "bar";
```

We can then retrieve that value in a later session with:

```
my $stuff = $session{"foo"};
```

While our program appears to be storing and retrieving values in **%session**, it is actually retrieving them from the database using DBI—which means that, so long as we ensure each user has a unique ID, we can keep everyone's values separate.

Since we have what amounts to a hash that extends across sessions, how can we store information on which URLs we have visited and when? The easiest way is to use the URL as a key into **%session**, then store the last time the user visited the site. For example, we can store the URL with the following code:

```
my $document_uri = $r->uri;
$session{$document_uri} = time;
```

We want to retrieve this information when determining whether a user has recently visited a particular link. In order to do that, we will modify `label_url` so that it expects a fourth argument, a reference to **%session**. This way, `label_url` will be able to retrieve session information about the URL in question. We create the reference by preceding **%session** with a backslash (**\%session**) before passing it to `label_url`. We then dereference the copy of **%session** as follows, at the beginning of `label_url`:

```
my $session = shift;
my %session = %{$session};
```

The full code of a working version of `Apache::TagNew`, including the `label_url` subroutine, is in Listing 3.

Listing 3.

The rest of `label_url` is largely the same, except for a portion in the middle where we test to see if the URL begins with a slash (/). We must be sure to store and retrieve the same key from **%session**; otherwise, we will get false readings regarding when we last visited the URL. Since we store the URL based on `$r->uri`, which always begins with a slash and is relative to our server's root URL directory, we should retrieve the URLs in the same way.

We do this by getting the current URL and removing everything following the final slash:

```
$current_directory =~ s|^(\S+)/[\w.]+$|$1|;
```

What is left is indeed the current directory, to which we can prepend the URL:

```
$url = $current_directory . $url;
```

Now we can retrieve the session information about that URL, confident we are using the same set of keys for retrieval as we did earlier for storage. We retrieve session information about when we last viewed the file in question, turning it into a number of days relative to right now:

```
my $last_time = (time - $session{$url}) / 86400;
```

Then we retrieve the modification timestamp of this file, by prepending **\$r->document_root** (the full path name leading to each file on the web site, normally invisible to users) to the file. We can easily determine its modification date:

```
my $full_filename = $r->document_root . $url;
my $ctime = -M $full_filename;
```

Finally, we compare **\$ctime** (the number of days since the file was modified) with **\$last_time** (the number of days since the user last saw the file). If the former is smaller than the latter, we add the label:

```
if ($ctime < $last_time)
{
    $label = "<font color=\"red\">New!</font>";
}
```

This module seems to do a good job of labeling new documents on a user-by-user basis. As long as users enable cookies, they should be able to get an accurate reading of which files they have not seen in a long time.

Conclusion

For a medium that is supposed to adapt itself to our own needs, the Web is surprisingly primitive—for instance, in the way “new” documents are labeled on web sites. This month, we have seen how `mod_perl` allows us to personalize our site a bit more, showing people what is actually new from their perspective, rather than from the webmaster's perspective. I hope you also noticed how advanced some of these tools have become; with a little more than 100 lines of Perl code, we were able to make a substantial change to our web server that had little impact on performance, but provided great benefit to our users.

Resources



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 later this year. Reuven can be reached at reuven@lerner.co.il. The ATF home page, including archives and discussion forums, is at <http://www.lerner.co.il/atf/>.

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

[Advanced search](#)

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

[Advanced search](#)

Focus on Software

David A. Bandel

Issue #64, August 1999

Open Business Management, gensig, xchat and more.

The distribution front appears to have settled down. All the major distributions now seem to have an offering with a 2.2 kernel and glibc 2.1, and work has started on the 2.3 kernel. I am looking forward to a journaling file system, but it will undoubtedly require a particular format when it is introduced. After transitioning from a.out to ELF, then from libc5 to glibc, I should be used to it, but it is still mildly annoying. It is probably a good idea though, as it allows me to get rid of the chaff. I also have a few packages that compiled under glibc-2.0.7 which will not compile under glibc-2.1, but I am hopeful these will be patched soon.

Open Business Management:

<http://www.aliacom.fr/obm/>

The particular version of Open Business Management I downloaded (the latest at the time) is not complete, but has most of the required functionality. The documentation (TODO file) points to much work yet to be done, but this package will have a great deal of potential once it is nearer to completion. Separate web pages allow you to insert, find or modify company, contact or deal information. The use of "themes" is a nice touch and allows anyone to tailor the look, although it is not as important as the missing administration module. It requires Apache, php3 compiled with MySQL, MySQL and a web browser.

gensig:

<http://www.geeks.com/~robfgensig/>

Need or want to have different signature lines? **gensig** will read a text file (one is included with **gensig** as an example) and go through selecting successive lines for each new outgoing mail message. The program looks for tag lines in files in various locations, including (first) the user's home directory, so each user can have his own tag-lines file, or the generic one can be expanded. It requires glibc.

xchat:

<http://xchat.linuxpower.org/>

IRC is not a program I use much. I personally find it too time-consuming and intrusive. I much prefer e-mail or a forum such as **wwwthreads** (below) to handle communication. However, **xchat** is a very friendly program for first-time IRC users. The program comes with several preselected servers programmed in, but can be easily changed. The program opens in two boxes: one for chatting, the other to manage connections. It requires gtk+1.2, glib-1.2, libgdk, libgmodule, libdl, Xext, X11, libm, libnsl, libdb, libpthread and glibc.

wwwthreads:

<http://www.wwwthreads.org/>

wwwthreads provides all the functionality of forums found on bulletin boards. Any number of forums can be created, and users may be designated administrators for boards. Posts can be public or private, and forum users can elect to have forum messages e-mailed at each posting or only for private postings. Administration, posting and reading are all done through a web interface, making administration simple. This program should be fast and powerful, capable of handling large message loads and thousands of messages due to its use of a MySQL server as the back end. It requires Apache, MySQL, Perl 5 and the Perl modules DBI, DBD, Data Dumper and Data ShowTable.

NetLED:

mars.ark.com/~mbevan/products/netled.shtml

Most systems have Ethernet cards with a link light on them. The problem is, the link light is on the card and can be seen only from the back of the system, an inconvenience at best. **NetLED** allows you to use the LEDs on the keyboard to indicate the status of the network as seen by **ifconfig**. By default, the "caps lock" key indicates send, "scroll lock" indicates receive, and "number lock" indicates carrier. This can be changed as desired via a configuration file. It requires glibc.

ntop:

<http://www-serra.unipi.it/~ntop/>

ntop is to the network as **top** is to your system. That statement may be selling ntop short. The **curses** version certainly shows a top kind of view of your network and is configurable just like top, but it can also be viewed with a web browser and has pages of colorful, easy-to-read data. This program tells you almost everything about your network. Be careful; this program presents a possible security vulnerability if non-privileged users are allowed access. Since it can see all network traffic and read passwords passed in the clear, it is in effect sniffing your network. It requires ncurses, libpthread and glibc.

stickerbook:

<users.powernet.co.uk/kienzle/stickers/index.html>

This sticker book is an excellent program for children 3 to 7 years of age. It is even easy to use for adults who have not yet mastered the GIMP. The program can be configured to have a large, easy-to-use pointer that is constrained to the program box. Three scenes are available, and the author is looking for assistance in creating a few more. It is about time we have something for little hands. My four year old has been logging in for almost a year, but could only play with the screen saver, and even that with some difficulty since it is a small icon and smaller pointer. Now I may never get a chance to use my own system.

xhanglider:

<http://plaza.harmonix.ne.jp/~redstar/>

xhanglider is a cute little nothing of a program that displays a hang glider flying around your root window while you work. Nothing but pure enjoyment—we all need a little diversion once in a while.



David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When not working, he can be found hacking his own system or enjoying the view of Seattle from an airplane.

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

[Advanced search](#)

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

Graphical Toolkits for Linux Programs

Patrick Lambert

Issue #64, August 1999

A brief look at several popular toolkits available for Linux.

Linux is now a reality—it is no longer just a hacker tool or a toy for students. As such, it needs the power of console programs ported to the graphical world. People want easy-to-use desktops with good-looking programs. This is why many programmers now turn to graphical toolkits.

While many toolkits are available, they all share some basic features. Since programming languages do not include built-in functions to make graphical widgets, you must use add-ons. Graphical toolkits are actually libraries which add functions to a programming language, allowing you to integrate a graphical interface to your program.

The main differences found between various toolkits are ease of use, graphical appeal, cross-platform portability and language. If, for example, you are experienced in the Tcl scripting language, you will probably want to use the Tk graphical toolkit. If you like Perl, you may pick Perl/GTK.

GTK, the Gimp Toolkit

The GTK toolkit seems to be one of the most popular. It is modern and easy to use. That library was made in C as the base for the GIMP, an image manipulation program. Now it is used by programmers around the world for all kinds of applications, including the GNOME desktop environment. The graphical interface looks clean and much like interfaces on other operating systems.

GTK is a good toolkit for writing applications in C, since it is a C library. The GTK toolkit is built on top of the GDK library, which is on top of GLIB. All three provide unique functions to programmers. Available functions include memory

handling, graphical components and widgets. GNOME also has its own extensions. GTK and GNOME are freely available software products.

The GTK library is available at <http://www.gtk.org/>. A tutorial on GTK is available at <http://www.gtk.org/tutorial1.2/>.

QT from Troll Tech

The QT toolkit was created by Troll Tech, a software company in Norway, and is used in the KDE desktop environment. It is written in C++ and used by programmers worldwide. The QT library began as a commercial product, but now Troll Tech has released a free version under an open license. Similar to GTK, it has the same kind of widgets, including labels, entry boxes and text fields. QT would be a good choice if you write applications in C++. The QT library is cross-platform, and the graphical interface of programs using it will compile without changes in both UNIX and Microsoft Windows.

The QT widgets look very much like GTK's widgets, as well as those of other operating systems.

The QT library is available from <http://www.troll.no/>. A tutorial on QT is available from www.troll.no/qt/tutorial.html.

wxWindows

wxWindows (w for Windows, x for X Window System) was created at the University of Edinburgh as a cross-platform toolkit. wxWindows is a C++ framework that allows you to write graphical applications. You can write your code once, then compile it under one of the many ports of the library. It currently runs under Microsoft Windows, Macintosh OS, Motif under UNIX and GTK. There is one library per platform, all providing a common API.

wxWindows is a free product, under a license similar to the L-GPL. Using it, you can write both commercial and free products.

wxWindows is available from <http://www.wxwindows.org/>. A tutorial is available from web.ukonline.co.uk/julian.smart/wxwin/hello.htm.

GraphApp, Platform-Independent GUI Programming in C

This toolkit is a favorite of mine. GraphApp is a C library that allows you to write simple graphical applications in C. It is a cross-platform toolkit, and will work on the Macintosh OS, Microsoft Windows, Motif under UNIX and Athena.

GraphApp supports a more limited number of widgets, but is truly easy to learn. You can learn how to make graphical applications in less than an hour.

A nice thing about GraphApp is that it compiles as a small static library. This means you can compile your programs with the library linked in them without increasing the size of the binary much, and the user will be able to run it without installing the toolkit.

The GraphApp toolkit is available at <http://www.cs.usyd.edu.au/~loki/graphapp/>. A tutorial for GraphApp is available at <http://www.cs.usyd.edu.au/~loki/graphapp/tutorial/>.

Motif, the Standard

Motif has been the standard graphical toolkit for years on UNIX and other platforms. It is a commercial standard and has its own look. Motif is the base for the popular CDE desktop environment, also a standard on many commercial UNIX systems.

On Linux and other open systems, developers have made a free Motif clone called LessTif. LessTif is source compatible with Motif and available under the L-GPL. Motif and LessTif offer cross-platform compatibility among UNIX systems. While Motif code will not work on most non-UNIX systems, many commercial UNIX systems come with Motif libraries. Also, Motif has the advantage of having passed the test of time.

LessTif is available from <http://www.lesstif.org/>. LessTif documentation is available from www.lesstif.org/Lessdox/lesstif.html.

While I have not covered all existing toolkits, I have briefly covered the most popular ones. Most programmers are concerned about two things: graphical look and portability. GTK and QT are probably used the most in the Linux world, mainly because of the GNOME and KDE desktop environments. Users want a desktop that will provide all utilities using the same graphical look. I use both GTK and GraphApp, but this is a personal choice which every programmer must make for himself.

The GNOME web page is at <http://www.gnome.org/>. The KDE web page is at <http://www.kde.org/>.



Patrick Lambert is currently a student in Computer Science at the University of Montréal. He has been using various UNIX and Linux systems for 5 years, doing

software development and systems administration. He can be reached at drow@darkelf.net.

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

[Advanced search](#)

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

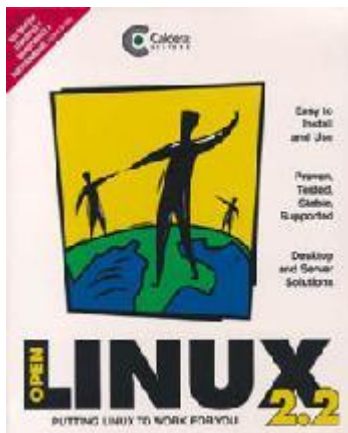
Advanced search

Caldera OpenLinux 2.2

Jason Kroll

Issue #64, August 1999

OpenLinux comes with a number of productivity packages including Corel's WordPerfect 8 and StarOffice v.5, along with the typical Linux utilities and applications (even games) which are all incorporated into the menuing system of the K Desktop Environment.



- Manufacturer: Caldera Systems
- E-mail: info@caldera.com
- URL: <http://www.calderasystems.com/>
- Price: \$49.95 US
- Reviewer: Jason Kroll

OpenLinux 2.2 is Caldera's latest "Linux for Business" distribution, a professional, productivity-oriented package that is highly functional, stable, secure, and especially easy to install and administer. OpenLinux comes with a number of productivity packages including Corel's WordPerfect 8 and StarOffice v.5, along with the typical Linux utilities and applications (even games) which are all incorporated into the menuing system of the K Desktop Environment. Also included is the Caldera Open Administration System (COAS), which is an easy-to-use graphical interface for dealing with configuration and

administration issues (similar to a Control Panel on the Macintosh, but aimed at network administration).

Installation

OpenLinux 2.2 is installed from an autobooting CD-ROM by way of Caldera's graphical installation program Lizard (Linux Wizard, that is). Provided your computer has a CD-ROM drive, autobooting from the CD-ROM is achieved by pressing **delete** during startup and selecting the CD-ROM drive as the first boot device, or you can start up the installation program from within the particular OS which comes by default on most PCs (MS Windows). OpenLinux's installation CD boots immediately to the graphical interface, which *is* actually easy to use. Once the Caldera graphic comes up (dancing people on part of a globe, presumably happy because Linux installation is so easy), Lizard will present a boot prompt, and if nothing is typed, will begin probing the hardware. Lizard's probing techniques are fairly thorough and if they do cause the computer to hang (with the diversity of hardware available, problems are bound to come up on some systems), typing **install er=cautious** at the boot prompt will tune down the probing and the installation should continue smoothly.

The graphical installation process, once begun, is extremely simple and consists of a few questions and answers. The first question is which language you would like your installation instructions in; the choices are English, German, French and Italian. Strangely, Lizard will not let you change your mind on this subject, so one is advised to be truthful. Once the language is chosen, Lizard configures the mouse; the mouse should work with the settings derived from Lizard's probe, and changes to the configuration could cause a malfunctioning pointer.

Lizard presents several partition options: entire hard disk, prepared partitions and custom. The first option simply prepares the entire hard disk for use with Linux; the second is used for situations in which one has already partitioned a drive for Linux. The final partition option is for "experts only" and is what the typical Linux user is used to: manual editing of partition information (gasp). All of this is quite simple. The only problem I noticed was that Lizard would sometimes ignore my swap partition, and I would have to go back and declare my swap partition again (just a simple back-and-forth clicking, like on a web browser). According to the documentation if you boot from Windows, PartitionMagic is invoked automatically to build the partitions for you. Once the partitions are formatted, selection of installation packages is made.

Unlike many Linux distributions, Caldera OpenLinux does not require a user to spend time picking and choosing particular packages. Four installation options are offered: minimal (server), recommended, recommended plus commercial

(including WordPerfect and Star Office) and complete install (over one gigabyte of Linux software). I tried all four sizes, and I recommend one of the latter two, preferably the complete install if you have the space.

A clever technique of Lizard is that installation of packages proceeds while the user is still answering questions and configuring the system. Right after package selection has begun, the questions on keyboard layout, video card, monitor and networking are raised. Keyboard layout is just a typical straightforward selection (although "Finnish" is misspelled as "Finish"). The video card selection offers a probe function, which I recommend. On the test machine, Lizard correctly diagnosed the video card, while the probe discovered the correct amount of video memory. Monitor selection consists of choosing among over 1,700 monitors from a database. Obviously, some monitors may be missing from the list, but any monitor which is agreeable to VESA standards should be easily configured.

After the monitor has been selected, Lizard shows possible resolutions for running the X Window System and KDE. To test a particular resolution, simply select the resolution and click the **test** button. One difficulty I had is that Lizard showed a KDE screen instead of the standard X test pattern, which made it difficult to tell if I had a *perfect* setting. Still, it is quite an easy procedure: point and click until the test screen looks right.

Finally, just set the root password, add a few users, perhaps configure networking, and then all is ready to go!

OpenLinux and the K Desktop Environment

Caldera selected the K Desktop Environment as the interface of choice for its distribution, and upon startup, OpenLinux launches right into it. If you feel uninclined to type in your user name, pointing and clicking from a list of users is an option (regular users' icons are just heads, but root gets to be a conductor). However, you *will* have to type in your own password—there is no point-and-click solution to this problem (yet).

The first login from any user will bring up KDE's configuration program, which asks a few questions as to which theme and icons the user would like. There are some typical KDE themes from which to choose (KDE default, Macintosh OS, BeOS and Windows). As for icons, you can choose floppies, CD-ROM, printer and even a Caldera Systems icon for your desktop, linking you to their home page, if you are really enthusiastic about Caldera or think you'll be needing technical support. One mysterious issue of this icon selection is that unless you move the configuration window, you will not see any icons appearing on the desktop, but that is the only problem I found with KDE.

Caldera clearly put a lot of effort into their particular configuration for KDE, because the menuing system is neatly organized and includes a great many applications, utilities, networking software and other programs (even games). It is quite impressive to click on the K icon ("Where do you want to go tomorrow?" it asks) and see everything laid out so clearly, without having to do any manual configuration. In fact, the sight of such a selection of productivity, networking and development software all configured and ready to go on one system could be one of the most persuasive arguments in favor of running Linux in a business environment; that is, when we try to convince non-Linux users to give our OS a chance.

Caldera Open Administration System

For some reason, an aversion to shells and text-based interfaces has developed among modern computer users; the preference is for GUIs these days. COAS is Caldera's solution for network administrators who prefer a graphical approach. COAS is an interface for dealing with issues of network administration and is set up so as to keep track of administration changes in such a way that one can use COAS when one chooses and traditional Linux commands when one is so inclined.

Issues

Caldera OpenLinux 2.2 is very close to requiring minimal technical knowledge; however, some problems can arise. The first is that LILO (Linux Loader) often has trouble installing on partitions with more than 1024 cylinders. This is easily remedied by appending **linear** to the general section of the `/etc/lilo.conf` file, but since one might expect most modern hard drives to have more than 1024 cylinders, it seems strange that Caldera did not automate the accommodation for this problem. Still, I had some trouble making a successful coexistence with the BeOS. The other problem is that because OpenLinux 2.2 aims to be secure, authority issues can arise when trying to restart X and KDE after one has shut down the windowing system which comes up at startup. When a user logs into a console, the greeting message says to type **startx** or **kde**. Neither of these commands was functional on the test machine except for root, and then only after I made changes to the `.xsessionrc` file. In a business environment, this could be confusing, though the solution may simply be to reboot. Also, StarOffice must be configured before it can be run, which could pose difficulties for certain people. Finally, for some reason, Caldera did not include the immortal editor **pico**--I had to boot up Emacs all the time. I am *sure* there is a simple remedy for this problem.

As far as being up to date, OpenLinux 2.2 runs on the Linux kernel 2.2.5. The 2.2 kernel is a fantastic production and is especially improved in the areas of networking, interoperability with other operating/file systems, parallel

processing, backup and RAID and hardware support, things which are especially important in a professional environment. Also, the OpenLinux 2.2 distribution is up to date on its numerous libraries, while maintaining backwards compatibility for glibc 5-based applications.

Support

One of the only arguments supporters of Microsoft could make against Linux is that consumers would not use Linux because they want total product accountability from one particular firm (in essence, users of free software would have no one to call for support). However, Caldera stands behind its product with a 90-day/five-incident warranty and keeps a database of potential bugs and fixes on its web site. Support forms are available on-line, in case one has a problem without a readily visible solution, and Caldera is relatively quick to respond. For businesses and individuals who cannot wait for technical support, there is a full 24x7 fee-based support system with guaranteed one-hour response. Also, Caldera insists it is working to ensure Y2K compliance, though Linux users have traditionally not been very concerned about this.

Conclusion

OpenLinux 2.2 is designed specifically for the business community, and although it would be quite good for productivity-minded home users, it is highly tuned for business use. Businesses should find everything they need right here, including single-party accountability for technical support. The system appears rather secure, and passwords are shadowed. Also, I noticed I could not log in as root via TELNET (which is good for security).

Ultimately, this distribution is evidence of how easy Linux can be to install, and how commercially and professionally viable it is. Home users who like to hack might prefer another distribution, but for businesses, this appears to be *the* solution.

Jason Kroll has been copy editing articles for *Linux Journal* for about three years. He is now employed here full-time as our Technical Editor/Product Reviewer. He can be reached via e-mail at info@linuxjournal.com.

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

[Advanced search](#)

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

[Advanced search](#)

Helius Satellite Router

Phil Hughes

Issue #64, August 1999

The Helius Satellite Router is a computer system designed to offer an Internet connection with a high download speed to a local computer network.

- Manufacturer: Helius
- E-Mail: info@helius.com
- URL: <http://www.helius.com/>
- Price: \$2499 US plus dish and installation
- Reviewer: Phil Hughes

The Helius Satellite Router is a computer system designed to offer an Internet connection with a high download speed to a local computer network. It is based on a small computer running Caldera OpenLinux that includes a receiver card to connect to a DirecPC dish. You connect the system to your local network via Ethernet. The system then routes outbound traffic to your ISP via a 56K modem. Inbound traffic is received via the satellite dish and routed back to the appropriate host on your local network.



Figure 1.

All your local network does is treat the Helius box as a gateway. It then handles all the interaction to ensure your traffic is routed properly. The result is much

like an ADSL connection to the Internet with IP masquerading thrown in, so you can connect a whole network to the Internet with only one IP address.

The system works just the way an experienced Linux hacker would expect. It takes advantage of Linux's built-in masquerading and demand-dial capabilities. You connect the internal modem to a phone line, the receiver card to a DirecPC dish and the 10BaseT Ethernet port to your local network. When a system on your local network requests something from the Internet, the Helius box calls up your ISP, sends out the data and waits to receive data via the dish. When the data is received, the Helius box then routes it back to the requesting system on your local network.

You may be thinking that this system is the same as the one offered by DirecPC—it is not. The standard DirecPC system offers connectivity for a single PC, not a network. That is fine for a single home user, but not for a multiple-system network.

Installing the System

Helius offers installs for under \$200, but I elected to do the install myself. After all, why not get the full experience? Helius sent me a dish and I went out and bought a dish installation kit for \$60. It included the necessary cables, connectors and even a compass to aim the dish.

I hooked up the dish and did a preliminary aim based on numbers furnished by Helius. I then hooked up a keyboard and monitor to the Helius system. (They don't come with it and are needed only for the dish install.) Armed with a primitive but effective program and an assistant, I got the system up and running. This process took a few minutes and mostly consisted of me yelling out numbers to my assistant while she moved the dish.



Figure 2.

Once up, I connected the system to my network and the phone line and was off and running. I set the gateway addresses on the machines on my network to

correspond to the address of the Helius box and started talking to the Internet. That was it. The Helius box dialed my ISP and routed packets without incident.

As far as performance, I had trouble finding sites that would test the capabilities of the system. Helius claims transfer rates of up to three megabits per second. I saw rates of 37KB per second doing an FTP access to SSC's FTP server and a 31KB per second transfer rate to the same server while simultaneously saving Usenet news. With my usual transfer rates at 4KB per second on my 56K modem, I am rather pleased with these new rates.

Who Needs It?

While this system is not the answer for everyone, it certainly has an audience. If you are a small company or school that needs an Internet connection with high download bandwidth, this is an ideal system. In an area where ADSL is available, you need to evaluate the pros and cons of DSL versus this wireless technology. If DSL is not available in your area, this system seems like an easy win.

Handling tens of users is easily attained with a minimal setup and monthly cost.

Costs

The system costs \$2499 for up to 30 concurrent users. This includes 90 days of support/maintenance. Extended support is available in the US for \$299/year. Add to this the cost of the dish, about \$300, and if you elect to, about \$180 for installation.

In addition to the cost of the system, you need to pay for monthly service from DirecPC and your ISP. DirecPC charges \$109.99/month for the first 200 hours of connectivity and \$.99 for each additional hour. ISP costs vary. What I see in Seattle is around \$20/month for the first 100 hours, \$.50/hour after that. Thus, for under \$200/month you can get 200 hours of on-line time with a fast download transfer rate.

Is the Helius Satellite Router for You?

This question boils down to, "do you need a fast download system for a network?" If so, the Helius Satellite Router is an ideal solution. It is capable of talking to a UNIX/Linux-, Windows- or Macintosh-based network, easy to install and performs well at a reasonable cost.



Phil Hughes is the publisher of *Linux Journal*. He can be reached via e-mail at info@linuxjournal.com.

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

[Advanced search](#)

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

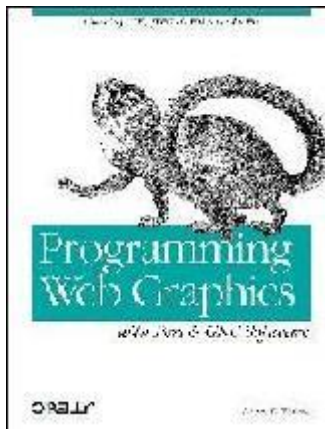
Advanced search

Programming Web Graphics with Perl & GNU Software

Michael J. Hammel

Issue #64, August 1999

Although most of the graphs seen on the Web these days use in-house developed tools, many of them started out using the same tools described by Shawn, such as ImageMagick and the GD library.



- Author: Shawn P. Wallace
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- URL: <http://www.ora.com/>
- Price: \$29.95 US
- ISBN: 1565924789
- Reviewer: Michael J. Hammel

About a year and a half ago, I started investing on-line. The first thing I did was research the stocks I found most interesting. Yahoo's stock section is quite good—many important-looking numbers and graphs. The graphs took me by surprise—I hadn't, at the time, thought about on-the-fly images. Much of the data used for the graphs is fairly static (one-, three- and five-year reports, for example), but it makes no sense to keep these graphs around permanently. It

would be an awful waste of disk space, and the data might change unexpectedly.

I am now far more familiar with image development in general, but until recently I was still fairly unclear on how dynamic graphics were built for the Web. I was, that is, until *Linux Journal* sent me a copy of Shawn Wallace's *Programming Web Graphics with Perl and GNU Software*. Although most of the graphs seen on the Web these days use in-house developed tools, many of them started out using the same tools described by Shawn, such as ImageMagick and the GD library.

As the name implies, Shawn's text focuses on writing tools using Perl. Almost all of the tools he talks about in detail have a Perl module, making it easy to integrate them into your scripts. Other tools work from the command line, so calling them from Perl is still fairly easy.

The text opens with some basic network graphic issues and a comparison of the most popular (web-supported) graphic file formats. The comparisons are quite good, providing information on header formats and some pointers on when and where these formats are best used. One interesting bit was on JPEG (Joint Photographic Experts Group). Apparently, the compression techniques in JPEG work well for the human eye's perception of color, but tend to distort text much more than GIF (Graphic Interchange Format). I didn't know that before reading this, and now understand that screenshots of dialog windows are probably better saved as GIFs. The trade-off, of course, is in file size. No format is perfect.

A couple of good and bad things can be said about the book. First, it sticks to well-known and published standards—in other words, it sticks to HTML 3.2. That is a good thing, but unfortunately it doesn't talk about HTML 4.0; for example, there is no discussion of style sheets. It also doesn't give much space to Netscape or Internet Explorer extensions. Deny it if you want, but either of these two browsers is the likely tool of choice for the majority of users. At a minimum, a table showing some of the differences or browser-specific extensions would have been useful.

Back on the good side, Shawn does an excellent job of introducing and defining APIs (Application Programming Interfaces). Throughout the book, he first discusses a graphic library interface, its API, and then follows with meaningful examples. Well, most are meaningful. That biorhythm example in Chapter 6 seemed just a little too *Austin Powers* for my taste. I lived in the 60's, and truly hate going back there.

Most chapters also include in-depth function references. This is highly useful information and somewhat unexpected, considering that each chapter starts with some straight, non-technical talk about what to do with these libraries. Again, Shawn goes the extra mile to provide the most useful technical information.

Two very good chapters are Chapter 5, “Industrial Strength Graphics Scripting with PerlMagic” and Chapter 7, “Web Graphics with the GIMP” (GNU Image Manipulation Program). The former is extremely useful, even if you're not doing web graphics specifically. It includes a very extensive description of the PerlMagic interface to ImageMagic.

The GIMP chapter is too sparse for experts and too technical for new users. However, it does provide information on the various animation options, much of which I didn't know. The section on Perl-Fu is good, but Perl-Fu requires Perl 5.005, which most Red Hat 5.2 users don't have. (SuSE, Debian and other distributions may have this—you need to run **Perl -V** to find out.) I find it a bit annoying that so many application-level tools make requirements of libraries and low-level utilities not available with standard distributions. This shouldn't be surprising with an environment as young as Linux, but it is still annoying. Even so, the Perl extension to the GIMP is a prize possession and shouldn't be overlooked.

The Perl-Fu interface comes with several Perl modules. The GIMP module allows access to plug-ins and filters via the standard plug-in interface, **gimp_run_procedure**. The **GIMP::Fu** module abstracts this so you call plug-ins by name. At least, this is the impression I got after reading this chapter. Check out Appendix C, which is a very good reference for writing Perl scripts for the GIMP.

The most confusing paragraph in the entire book came in the chapter on the GIMP. Here, in a footnote, Shawn makes a valiant attempt to explain the differences between plug-ins, script-fu scripts and Perl scripts. It isn't an easy read, but it was a nice try. I noticed one error in chapter 8, “Image Maps”, in reference to the GIMP: there *is* an Image Map tool for the GIMP now. I suspect the reason Shawn missed this is simply a matter of timing. The Image Map plug-in probably arrived after Shawn finished writing this book.

In general, the text makes a fairly poor attempt at being platform-independent. Occasional references are made to Microsoft-based tools, but many (perhaps most) of the tools discussed are UNIX-specific. I think O'Reilly may have wanted this more than Shawn did, but in either case, they probably should have stuck with UNIX only.

The chapter on animation is good, discussing the topic from a strictly programming perspective. Many of the techniques covered, such as splitting an image, often work better using interactive tools such as the GIMP. Nevertheless, much animation work is repetitive, and having programming tools for this process obviously helps. This chapter does a good job explaining the hows and whys of animation programming from a web developer's point of view.

One thing I completely disagreed with was the suggestion in chapter 10 of embedding ASCII art in HTML **ALT** tags. If you want to use ASCII art, as a convenience to users of text-only browsers, set the **ALT** tag to the appropriate text and link the tag to a separate page containing the ASCII art. Don't embed it in the **ALT** tag. If you do, you'll hate yourself later when you have to go back and edit that HTML file by hand.

Summary

Overall, I was quite impressed by Shawn's work on this subject. My very few complaints are:

- Black and white images made it difficult to see artifacts in side-by-side image format comparisons.
- Chapter 2 gave Perl code first, *then* mentioned modules described in later chapters. I thought I'd missed something until I read past the code.
- Chapter 3 mentioned alignment with "text flow" but did not define what that means. (It refers to alignment with window edges, table or cell frames, and so forth.)

These problems are minor and stood out only because I was specifically doing a review. They don't distract from the main purpose of the book, which is to guide developers in generating web graphics programmatically. The real meat of the book—the in-depth API descriptions—far outweighs any of the smaller issues.

After reading the book, I must say I was initially disappointed with the lack of reference to some tools, such as WhirlGIF for animations. Then I thought more about the title of the book and how it *does* mention "GNU" specifically. If you live and breathe only GNU, this text is perfect for you. If you are interested in GNU as well as other options, it will fill only part of your needs. Still, it is a fairly solid part, and definitely a great place to start when learning to program web graphics on the fly.



Michael J. Hammel (mjhammel@graphics-muse.org) is a graphics artist wannabe, a writer, and a software developer. He wanders the planet aimlessly in search of adventure, quiet beaches and an escape from the computers that dominate his life. He also likes squirrels and the letter "M". For no particular reason.

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

[Advanced search](#)

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

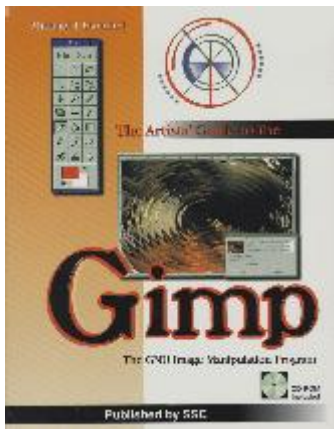
Advanced search

The Artists' Guide to the Gimp

Syd Logan

Issue #64, August 1999

The GIMP is very closely related to Adobe Photoshop in terms of purpose and functionality, and some would say it is superior (one can't argue this in terms of its cost).



- Author: Michael J. Hammel
- Publisher: SSC
- E-mail: info@linuxjournal.com
- URL: <http://www.ssc.com/>
- Price: \$40 US
- ISBN: 1-57831-011-3
- Reviewer: Syd Logan

The GNU Image Manipulation Program (GIMP) is a freely distributed program suitable for tasks such as photo retouching, image composition and image authoring. The GIMP is very closely related to Adobe Photoshop in terms of purpose and functionality, and some would say it is superior (one can't argue this in terms of its cost). The GIMP represents a significant contribution to Linux in its quest to become a viable desktop platform, because for many desktop users, an image editing program such as the GIMP is considered a “must have”

application. A book such as *The Artists' Guide to the GIMP* is needed for programs like the GIMP to reach the widest possible audience, and so Michael J. Hammel, the book's author, and SSC have contributed to the success of the GIMP, and indirectly, to Linux as a platform by bringing this book to publication. I hope this is the first of many such books to follow.

Overview

The first impression I had upon receiving the book was its extensive use of color artwork and the quality of paper upon which it was printed. Having written a book on image processing, I know how expensive it can be to print books in this manner. Often, to reduce printing costs, a publisher will opt to print the color images together as a series of color plates bound somewhere near the center of the book. The difficulty with such an approach is it forces readers to skip back and forth between the text being read and the supporting illustrations, which isn't always convenient. SSC did readers of this book a favor by opting not to take the cost-saving approach; as a result, the book is full of color images, each located within a page of the relevant text.

The book begins with an introduction that discusses in detail issues such as minimum hardware and software requirements of systems running the GIMP. Much of the information could probably have been summarized neatly by a table or two that divulged the minimum and recommended amounts of memory, disk space and so on needed to run it. In the video requirements section, I sensed some minor confusion with regard to X11. I believe the graphics primitives used by Xt/Motif and Gtk+/Gdk are essentially the same (contrary to the author's claim), as both reside above, and make use of, Xlib. As a result, the graphics performance should be nearly identical, regardless of the toolkit used. The whole point is somewhat moot anyway, since the GIMP is not and never will be a Motif program.

Another source of minor confusion was the discussion of the X Shared Memory Extension. This extension can be taken advantage of by the GIMP only when both the X server and the GIMP are executing as processes on the same machine. This is the true reason an X terminal (for example, the NCD model that was mentioned in the book) could never take advantage of the extension—if an X terminal is involved, the server and the GIMP client are separated by a network connection, effectively executing on separate hosts. Also, if users are running the client on a UNIX/Linux host connected to an X server running locally on a desktop UNIX/Linux host which supports the shared memory extension, the GIMP still cannot take advantage of the extension, because a network connection separates the client and the server from each other.

Somewhere in Chapter 1, the author mentions how to start the GIMP from the command line by typing **gimp**. The **--display** argument is mentioned, but none of the other command-line arguments (**-h**, **-v**, **-b**, **-n** and so forth) are mentioned and neither is the fact that you can type **gimp -?** to obtain a usage report. I think a detailed book on any program that can be run from a command-line should cover each of the supported command line arguments in detail.

Most Linux users can skip the software requirements section, since the GIMP will be provided in a ready-to-run configuration with their Linux installation. For users of other systems (such as Solaris) or those of you wanting to get the sources and build the GIMP yourselves, helpful information is provided near the end of Chapter 1 to get you started.

Chapter 2, "GIMP Basics", was misnamed. A better title might have been "Image Basics". The chapter is short and provides a variety of information, including a basic discussion of raster graphics, color spaces, image and pixel organization, file formats and brief introductions to the GIMP's selection tools and layers. The discussion of image file formats was okay; the discussion of color spaces and pixel organization less so, but still adequate. Since the rest of the chapter dealt mainly with image and color space issues, the discussion of scanners seemed out of context. The same can be said regarding the discussion of layers and selection tools.

Chapter 3 is essentially a tour of significant portions of the GIMP user interface. It starts by listing various menu items available. Readers will want to skim this section in order to get an idea of how the menu items are organized, but plan to refer back to it later as a reference. I strongly suggest you launch a copy of the GIMP, open an existing image or create a new one, and play along as you read this chapter, trying out each menu item as it is discussed. The chapter continues with a discussion of some of the GIMP's dialogs, providing screen shots to illustrate each dialog. Even so, it is a good idea to practice bringing each dialog up; this will help you get savvy with the user interface much quicker. The chapter ends with a tutorial, which you can probably skip if you followed my suggestion and experimented with the GIMP as you read.

Chapter 4 covers the tool box, which is the main application dialog of the GIMP. Each button in the tool box is introduced and explained. I think this chapter, like Chapter 3, is a good one to skim and then return to later after the book has been read more completely. It has a reference-like organization and some of the terms introduced in the chapter may not be entirely clear until more detail provided later in the book has been read. This is especially true for those readers new to the GIMP and inexperienced with Photoshop or image manipulation programs in general. Chapter 4 ends with a "tutorial", which to

me seemed more like a quiz and an invitation to play with certain features than a tutorial.

Chapter 5, "Selections", does a great job of describing the various selection tools available in the GIMP and presents many tips and tricks regarding their use. The tutorial at the end of the chapter was well-designed, far better than the one at the end of Chapter 4, I felt. Chapter 6 presents a similarly detailed treatment of GIMP layers and channels. Another well-designed tutorial follows the material presented in Chapter 6. I think much of the power and overall coolness of the GIMP derives from features described in these two chapters, so you will want to study them carefully.

Chapter 7, "Colors and Text", discusses tools that can be used to perform image enhancements such as color balance changes, contrast enhancements, brightness modifications and histogram stretching. A concise description of the text tool dialog is presented, and the chapter finishes up with a look at Script-Fu logos. After a short description, each of the 21 logos supplied with the GIMP is illustrated. These logos look very cool, and I was compelled to try them out. I ran into font problems with most of the logos; this problem should have been anticipated by the author and help provided. I went to the index to see if there was an entry for "font" or "fonts" (a topic discussed a few pages earlier in the section entitled "Text Tool Dialog"). None was found. Scanning the table of contents, I found Appendix C, "Adding Fonts to the System", but no information to help me with my problem was provided.

By the way, a button in the Script-Fu logos dialog that could be used to bring up a simplified font selection dialog similar to the text tool would be a great addition. Having to type in the name of the font and the font size seems like a rather crude way of doing things.

Chapter 8 covers drawing tools including pens, brushes, pencils and airbrush tools. The discussion seemed complete; however, the explanation of airbrush tool controls would have been a bit clearer with some example illustrations. Palettes (both indexed and RGB) and the tools that support them are discussed, and the chapter ends with a look at GFig, a plug-in intended to enhance the drawing capabilities of the GIMP.

Since I am running out of space for this review, I'll just take a quick look at the remaining chapters. Chapter 9 is a short chapter that covers transformations such as cropping, rotation and the like. Chapter 10 covers gradients and ends with a good tutorial on the subject. Chapter 11, "Scanning, Printing and Print Media", ends Part 1 of the book. I don't have a scanner, and my printer was already configured for use with my Linux system, so the chapter did not hold

much interest for me (at least not yet—someday I will want to hook up a scanner, and this chapter is where I will start).

Following Chapter 11 is Part 2 of the book, which covers various Script-Fu effects. Each of the 12 chapters in this section is short, on the order of a few pages, and covers a specific Script-Fu effect. The chapters are more to be looked at than read; most of the content is provided by a series of before/after screenshots which show the discussed effect being applied to an image. I suggest using these chapters as a catalog of the effects that are available, and when you find one you like, use the example provided in the book as a starting point for your own creations.

Final Impressions

If you are serious about learning the GIMP, you will find a way to put this book on your bookshelf. If you are looking for something to teach you the GIMP via a series of examples or tutorials that build upon each other, you may need to look elsewhere. The book goes further than the typical “Learn XYZ in 21 Days” book and is a comprehensive reference to the GIMP and its operation that will prove itself useful to many readers. However, in the next edition I would like to see a chapter or two that takes readers through a detailed sample project, such as the creation of a complex image similar to the *Linux Journal* cover art Michael has produced in the past. This would help solidify the concepts presented throughout the book and give many readers a springboard for creating works of their own.



Syd Logan (slogan@cts.com) is a UNIX/Linux developer for Netscape Communications, working on the 4.x versions of Netscape Communicator. When he has free cycles, he tries to lend a hand with the GTK+ port of Mozilla (see <http://www.mozilla.org/>). He is the author of *Developing Imaging Applications with XIElib* and is currently at work on his second book.

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

[Advanced search](#)

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

Advanced search

Letters to the Editor

Various

Issue #64, August 1999

Readers sound off.

Letters

Typo in “Stop The Presses”

On page 10 of the May *LJ*, you write that Precision Insight is operating “With funding from Red Hat and XGI”. In fact it is “SGI”, not “XGI”, that's co-funding PI's work. SGI has also donated the GLX X Window/OpenGL integration source code to XFree86, and PI is using that code as one component of the OpenGL infrastructure they're developing. More on this is available at <http://www.sgi.com/software/opensource/glx/>. Thanks.

—Jon Leech lj@oddhack.engr.sgi.com

Oops, that was a bad one—my apologies to SGI for not catching this typo —
Editor

Intellectual Property in a GNU Environment

Having used Linux since before kernel 1.1.13, I've found a very long list of reasons why various businesses don't develop more software on Linux. Nowadays, the reason is never performance or reliability. One reason is well-known; that is, how does one go about marketing in an open-systems environment without giving away key rights or simply being unable to enforce those rights? Secrecy is usually less expensive than a court battle.

An issue recently mentioned to me is that of knowing how to interpret what a developer will owe to others. The example in mind is OpenGL in X. My interpretation is that for dynamically linked applications requiring some form of OpenGL-compliant display libraries, it is not an issue for the developer, but instead for the user (unless the developer ships the libraries with the

application). What I would like to see is a series of articles on intellectual property rights of developers, and to what extent these rights may affect marketing when a business (versus individual) must pay for using (not selling) various utilities and libraries used by a common distribution.

If, as a business, I use PPP utilities or generic NE2000 drivers but don't sell them, do I need to pay for them? If I use **gawk** as installed from my distribution, when do I need to pay for it? Can I use XFree as a business, without paying? Will my OpenGL application cause a liability to my end user who has Mesa? And so on. I would like to see a lengthy discussion worthy of showing to my employers. Thanks.

—Dan Stimits stimits@idcomm.com

As a start, read the article "Licenses and Copyright" by Michael K. Johnson in the September 1996 Linux Journal (issue 29). If you don't have a copy of this issue, remember that as a subscriber you have access to every issue on-line at <http://interactive.linuxjournal.com/> —Editor

Larry Wall

The feature article, "Larry Wall, the Guru of Perl" (*LJ*, May 1999), was both insightful and delightful. Marjorie Richardson was able to convey Larry's obvious passion, joy, intellect and humor.

It was also refreshing to see one of the key figures of the Open Source community unashamedly mention God. Yes, there are some of us using Linux who realize that the ultimate in "open source" is the Bible.

Larry may not have become a missionary, but he's used his God-given talents in a very good way.

—Bob Nelson bnelson@netcom.com

Actually, Larry was able to convey his passion, intellect, joy and humor himself. I just provided him with the medium in which to express it. I enjoyed doing the interview very much—he's truly a delightful man —Editor

Standards

I just finished reading the article "Distributions Take a Stand on Standards", and all I have to say is that all of this can be likened to Scotland when it was fighting the English (the movie *Braveheart* comes to mind). Microsoft (MS) is the English wanting to rule everything, and the Linux community is the Scottish. The clans of Scotland weren't unified and easily fell to the English. That is what is

happening with Linux today. The different distributions are the clans. No one is trying to unite them. Linux falls to MS because it can't stand up to MS. Linux may be winning some battles, but MS is winning the war, and it's easy to see why. Take a look at both armies. One is well-organized and well-structured. The other is made up of small clans, fighting for the same main goals, but refusing to unite for the greater good.

Caldera seems to be the William Wallace of Linux. It looks like they will be the ones to take the lead and unite the different distributions. I am going to stand with their banner in my hands—I am going to give my money to Caldera henceforth.

Linux needs a leader for standardization. Stampede Linux was quoted as saying too much of a good thing can be bad, but I have to say that Linux in its present state is a bad thing. I am a user for about five months now, and have bought both Red Hat 5.2 and Mandrake 5.3. I am very disappointed in both of them. I am not able to get my printer to work, because neither distribution works with a HP 722c. I know this printer is designed mostly for Windows, but programs are available that will make this printer work with Linux. Neither of the distributions support this program. If the major distributions want to pull together and make a standard for Linux, they need to put these small programs in their distributions and support them. If this was the case, Linux would gain more support from computer users in the world than it does.

—Troy Davidson clandaith@netzero.net

Layout

I love your high-quality magazine and look forward to each issue. Only one complaint: you've started putting a thick black border around the feature article pages. The ink sticks to my fingers and gets smudged all over the pages, making it much less appealing to read. Please get rid of the borders! I would also like Metro-X to get rid of all the black on their back page ad, which also smudges all over the place, but I guess that's another letter to be written to their marketing department. Or maybe your whole magazine is just too hot to read! Thanks.

—Walter Cooke wcooke@paragon.bm

We added the border to call attention to the feature section—makes it easy to find. The smudge factor did not occur to us and doesn't happen on the copies we receive. However, yours is not the only complaint we have received and we are talking to the printer about possible solutions. Perhaps we need to rethink this one. Thanks for your comments —Editor

Compounding Errors?

I will be the first to admit there are no absolutes in this world, but the comment offered by Shawn McMahon in the Letters section, *Linux Journal*, June 1999 issue concerning PCI modems not working in Linux, is absolutely wrong. Unfortunately, your response to his letter was likewise in error, thereby perpetuating this misinformation. You stated, "PCI modems are basically Windows-only." I realize you used a caveat in your response; however, as the (in my opinion) pre-eminent and "first-class" leader in Linux-related magazines/journals, you should do a little more research before making blanket statements that are inaccurate.

I am limiting my comments to one modem only for obvious reasons, but I must tell you that "some" PCI modems work equally well in Linux and Windows. For example, I recently installed a US Robotics/3Com V.Everything PCI modem in one of my computers, and have not experienced any problems with it whatsoever. I configured the modem to use com2 when operating in an MS Windows environment, and /dev/ttyS1 when using Linux. I connect to my ISP consistently at 48,000 bps (probably due to the antiquated wiring of our local telco) regardless of the OS I am running.

I enjoy perusing your journal immensely; in fact, I anxiously await its arrival every month. When it finally gets here, I scurry into my "library" and don't come out for an hour, much to the chagrin of the rest of the family. Kudos to all at *LJ*! Keep up the good work!

—Greg Bailey gsbailey@foto.infi.net

Sorry, everyone I talked to about Shawn's letter agreed he was right. With so much hardware available today, no one can know everything. Congratulations on finding a PCI modem that works for Linux and you —Editor

Linux in Schools

A Linux lab was constructed in December 1998 by a volunteer group of Linux users in Tucson, Arizona for Corbett Elementary school. All machines within the lab run only Linux, except for the district's server console—we were not given permission to remove Windows 95 from it, unfortunately. Even if it isn't the first Linux lab in a public school within the U.S., it's most likely the first in an elementary school (K-6). It contains 32 Compaq Pentium 90s, one AST Pentium 90 server, one 486 DX-33 Ethernet switch, one Gateway 486 DX/2-66 print server and one Compaq Pentium 200 MMX (district)—all running Slackware 3.6 and 4.0. The lab officially opened late January 1999, and the students love it. Some were having too much fun with XBill though, so we had to remove the gore. The entire district, which constitutes 105 schools, pipes out to the Net on

a single T1—not everything can be perfect. On that note, I'm the network administrator for this lab.

—James Daniel root@phreaked.net

Linux Standards and Distributions

I think your article “The Distributions Take a Stand on Standards” by Norman Jacobowitz (June 1999) does not prove a thing. No vendor in his right mind would state that they will not go along with existing and upcoming Linux standards and invent their own for “product improvements”. Not even Microsoft did so before introducing new proprietary software solutions.

At the moment, it's all about market share for a huge upcoming market and who's best in keeping up a good “front show” while thinking about how to get the biggest piece of the pie in the back office. What do you think about new software for Linux labeled as running on “Red Hat Linux”? Check out www.kai.com/C_plus_plus/download.html#intel_linux.

I think Linux needs an independent organization like the OMG for CORBA. At the moment, there is not even one “Linux Standard”. How do you explain to your customer which Linux configuration your software runs on? Some distributions know how to use this fact for their own benefit and could cause standards to go down the drain in a few years. Greetings from Germany.

—Roland Koeckel roland.koeckel@gmx.de

The Linux Standards Base, written about in “The Past and Future of Linux Standards” by Daniel Quinlan in that same issue, is striving to become that independent organization you are wanting —Editor

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

[Advanced search](#)

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

Advanced search

More Letters to the Editor

Word Processor Standards

Now that you have brought up the subject of standards, the issue of standards for word processors across the whole world of computers is one that needs addressing. It is already ten years overdue. The exchange of documents between Microsoft Word, Word Perfect, Apple and Applix is a catastrophe.

Thirty years ago, there was an organization called SHARE that through the umbrella organization of the Department of Defense, organized user companies from Exxon and General Motors to insurance companies, banks and the federal government to create standards for IBM.

Since GNU/Linux seems to have mastered the art of self-discipline on the herd of nerds who maintain the system, maybe we can raise the cry to start a movement to create a word processor interchange?

The Microsoft Word Standard is a mess and unacceptable. We need a standard for transferring documents between processors—import and export. Some organization with market clout needs to take the lead, but we can certainly publicize the problem to our benefit.

How do we start?

—Bob Stanfield, stanassc@epix.net

Your Review

I saw a link in *Linux Today* to your review of the UNIX awk and sed Programmer's Interactive Workbook.

Note: I am sending this comment to the reviewer and *Linux Journal* as well as posting at *Linux Today*.

I am the author of the *UNIX System Administrator's Interactive Workbook*. This is the original book in this series. I feel it would be helpful to clarify a few things. The intent of the books in this series is that they are accompanied by a book that is a more standard reference book. These books are designed for the beginner. Because this is new the approach will be fine tuned as we learn more about what is really needed for the student.

I have not actually read the *UNIX Awk and Sed Programmer's Interactive Workbook*, but I do know the intent of the series. I also cannot address

specifically the issues that are raised by the review. Basically the beginner is stepped through a process to learn the necessary skills. There is of necessity many things left out with the intent that the user will supplement the skills with other books. This may not be made clear enough in the text and accompanying publicity. I have made this clear in my text and publicity, but still people get confused on this issue.

My research interests include the process that the beginner uses to learn computers on their own. This is different, in my opinion, from how things are being taught in most Computer Science classes. Once this is better understood and utilized I think academic computer learning will be a lot more fun. It is like learning how to skate. You learn each skill as you go along. You then strengthen the skill and move on to the next skill you want to learn. This will take up a lot more text space and of necessity it will restrict what can be covered.

—Joe Kaplenk, jkaplenk@aol.com

Prediction of the End of UNIX!

Please accept this slightly tongue-in-cheek submission to your letters section of *Linux Journal*.

I predict that the Unix world will cease to exist on Tuesday morning, at 14 minutes and 8 seconds past 3 in January 2038 (in Sydney Australia). Why? Because at that time it will be 2147444048 seconds since 0 o'clock (you know what I mean), 1 January 1970. For those purists out there who know what I'm talking about already, the 39600 second (11 hour) difference is a timezone - Sydney to UTC - difference (and I'm too lazy to change the mickey mouse type test program to align to UTC).

How do we stop this?

The simplest way will be to change

```
typedef long    time_t;
```

in

```
time.h (or somewhere included below time.h)
```

to

```
typedef long long    time_t;
```

and then all routines associated with `time_t`. (If you don't accept that long long defines 64 bits, then substitute your favourite 64 bit storage class definition).

This will give us a few more years before the time flips over again.

No doubt this will take a few years to implement - perhaps a different set of routines, ie

```
time(long *) => ltime(long long *)
ctime(long *) => localtime(long long *)
```

etc and some macros in the time.h include file tree to help convert and warn existing users of these changes.

I realise that this problem is a long way off, but in the spirit of Unix (the socialist operating system as someone recently suggested), lets start the change now.

Just in case I have lost a lot of people, Unix time is based on the number of seconds since 00:00:00 UTC, January 1, 1970 and is stored as a long (signed 32 bit quantity), which means a maximum of $2^{31} = 2147483648$ seconds before the time overflows and we go to Friday Dec 13, 1901 (or at least that what my current Solaris box will do - again in Sydney Australia). By changing this storage class to a 64 bit signed quantity we get $2^{63} = 9223372036854775808$ seconds which should hold us for a few more years (approx 68 verses 292471208677 years).

Yes, we could make the change simpler and just go to a unsigned long storage type (given that the system call and associated routines don't make use of the high bit) but this just gives us another 68 or so years - ie move the problem to 2106 or somewhere nearby. And also, by the time this change starts most cpu/compiler ensembles will be either directly executing single op 64 bit instructions or will emulate them if not.

As a side issue, I want to congratulate Kyle Enlow (*LJ* letters June 1999). I was as annoyed as he was, when I read the Linux Certification article (*LJ* April 1999). We need Linux Certification as much as we need NT or Windows.

Regards

—Burn Alting, burn@comptex.com.au

Even before getting Linux

It's a great pleasure to say that *Linux Journal* is the greatest reason that I have installed Linux Redhat on my PC. I subscribe *LJ* about a year ago with my friend's help (using his credit card) and since getting my issue and reading all the good things about linux, I was hunting for it's installation CD. Well being a student with a limited budget, I manage to get Redhat 5.1 (stract marks all over) for a mere US\$3.00++ which is RM10.00 in Malaysia.

Reading all the greatness about it esp that Linux is not commercially oriented unlike Microsoft products which is too expensive for me.

I admit installing Linux was not an easy task for a beginner, without a proper reference maanual except for the readme and docs in the cd I manage to get it running Linux after 100s of trial and error procedures,

and after 1 month plus trying to understand how configure X windows properly, I got everything running well..

Up until now, I have no regrets for all the trouble installing it, and now after I got a proper book, which one of my lecturers gave to me, looking at how much interested I am in Linux. I find that it's so easy to install it..

Now I have customize it tailoring it, currently I am self teaching myself how to write Perl/C/C++, well learning Pascal, HTML and Java on my own was an example of I am determine enough to learn something even without a proper tools or resource.. thanks God my mother understand how much I love sitting infront of my computer all night and day just trying to better myself at it. The philosophy of linux itself thrill and move me. With everybody helping each other to better an OS for everyone's benefit is the most decent thing next to being a priest.

My life is full of excitement nowadays.. never dull like the days when DOS and Windows95 was ruling the OS for personal use. I hope someday, I would be a benefit to all those running linux, right now I just have to graduate.

P/S Linux made me an envy of my coursemates learning about Unix.. cos i tend to know more and answer the right question. :) Thanks LJ and Linus for the great OS. Keep it free so poor student trying to graduate like me can better themself.

—Desmond Lewis, joesonia98@yahoo.com

Rave

Thanks for such a great magazine. That's all, just thanks.

—Jamie Matthews, mattja@iglobal.net

Linux and PCI modems

Shawn McHahon (letters - June 1999) is partially right. There are PCI modems that won't work with Linux and there are PCI modems that WILL work with Linux, including the 2.2 kernel shipped with Redhat 6.0.

I'll bet there are a few heads scratching over the above statement. I fell into a working modem by sheer luck. Over a year ago I purchased A Sportster PCI 33.6 modem. It worked with Redhat 5.x. I just set the jumpers for COM2 and stick it in. Set the BIOS to turn off the internal COM2 and booting Linux. I had a 14.4 that also worked.

Because of some recent changes, I purchased a PCI 56k and sure enough, it refused to work. I took it back and got an external 56k.

OK, why did the 14.4 and 33.6 work. I gave some clues, which I hope you noticed. Yep, the PCI modems that can be manually configured for the comm port settings will work. PNP modems will not. When you can

manually set the modem, you are setting up the modem as if it were an external modem using a built in comm port. BTW, the 33.6 could be configured for PNP, I just never did it that way, being used to setting cards for the comm port that will be used.

So, if you can find (doubtful these days) a PCI modem that has configurable jumper settings, then it will work. Otherwise, like me, you have to use an external modem, which is a blessing because that frees up a PCI slot for a different card. You lose the comm port no matter what, so why not use the internal comm port and use the PCI slot for something else?!

—Mike Brown, vidiot@cm.nu

IP Bandwidth Management article in June 99 issue

This is in response to Jamal Hadi Salim's article titled "IP Bandwidth Management", in the June 1999 issue of *Linux Journal*. The article provides a clear description of the traffic control capabilities of Linux. It seems that some impressive packet scheduling work has been done for the Linux kernel. However, I take issue with some of the claims in the article. I'll address these in the following paragraphs.

Jamal states in his "Final Word", that "As far as I know, Linux is the most sophisticated QoS enabled OS available today." and that "I am not aware of any such functionality in Microsoft products." This came as a great surprise to me. Anybody who has been active in the IETF (the birthplace of RSVP, diffserv and most interesting QoS technologies) is aware of the work that Microsoft folks are doing in this area, in collaboration with folks from Cisco, Intel and numerous others.

To be a "sophisticated QoS-enabled OS" requires a lot more than a packet scheduler in the kernel. A QoS enabled OS needs to include at least packet scheduling, diffserv codepoint (DSCP) marking, and RSVP signaling. It is necessary to take a holistic approach to QoS. Allow me to elaborate - the purpose of QoS is ultimately, to enhance the experience of the user of networked applications, be they multimedia applications or other mission critical applications such as ERP, e-mail, etc. These applications suffer when there is congestion in the network anywhere between sender and receiver. The network starts with the sending host's network stack. However, in most cases, the congestion is not in the sender's stack, rather, it is somewhere deeper in the network. To provide QoS in the sending stack only and not in a true end-to-end manner is analogous to putting an HOV lane in the entry ramp to a congested highway. It doesn't make my commute to work any better - it just gives me quicker access to the congested highway.

How can hosts contribute to true end-to-end QoS? A number of mechanisms are available. Primarily, these fall into the realms of marking and signaling. Marking is the act of marking packets for certain behaviour in the network. This can be done by marking a DSCP in IP packet headers. Diffserv enabled routers in the network prioritize packets based

on the behaviour specified in the packet headers. An additional form of marking is 802.1p. This marking is achieved by adding to a packet's MAC header when it is submitted to an 802 LAN. Many switches prioritize packets based on the 802.1p mark. Signaling mechanisms include, at layer-3 - RSVP. At layer-2, ATM signaling, proposed extensions to DOCSIS for cable modems and a myriad other link layer specific signaling protocols. Signaling protocols are used to reserve resources for certain traffic flows in the network, to assist the network in recognizing traffic from certain users or applications, to provide admission control and feedback to applications and to coordinate QoS mechanisms among separate network elements. Recent work in the IETF has led to the integration of RSVP signaling with diffserv packet handling. This marriage of technologies enables the benefits of RSVP to be realized without the scalability concerns that impeded its early deployment.

Providing true end-to-end QoS in a host OS requires support for both signaling and marking mechanisms as well as traffic shaping. A host OS needs to provide support for these mechanisms in an integrated manner, as opposed to isolated 'boxes' of functionality. The mechanisms need to be able to respond to standard network policy controls. Finally - the host needs to provide end user applications and traffic management applications a simple and cohesive set of APIs that pull together all the disparate QoS mechanisms. Windows 2000 does so by providing a simple extension to the Winsock API which coordinates RSVP signaling, DSCP marking, 802.1p marking, ATM signaling (and additional layer-2 signaling mechanisms) and packet scheduling. As such, I claim that - while Linux may have the most sophisticated packet scheduler, it is Windows 2000 that provides the most sophisticated, ready-to-use QoS support.

—Yoram Bernet, yoramb@microsoft.com
QoS Development Lead, Microsoft

Re: IP Bandwidth Management
RE: Yoram Bernet's letter.

The Linux 2.2 is shipping, now, and the QoS work has been shipping since at least the fall of 1998. Windows 2000 is not yet shipping, nor are a dozen other products. So, with the exception of whatever W98 has, the rest is really vapourware as far as the end user is concerned.

However, when W2000 ships, I'm sure that its solution will be complete, while what is shipping with Linux is rough and lacks a good user interface.

Michael Richardson, mcr@solidum.com

At The Forge Factual Error - Hidden Fields - Don't trust them

To: jlarsen@ajtech.com

You're right about not trusting hidden fields. I'm surprised that I told people not to worry about hidden fields, or the origin of a form, since I'm

aware of the potential security risks involved. Thanks for reminding me of this; I hope not to forget it when writing future columns.

Using MD5 to verify the contents of a hidden field is a rather clever idea.

I'll take a look at MIME::Lite for sending mail with attachments. It sounds like a useful module to use in my own work, as well as in future ATF articles.

As for more articles about mod_perl, I certainly expect to do so in the coming months. I find mod_perl to be increasingly fascinating and useful, and think that others will benefit from trading ideas about it.

Thanks for writing, and please let me know if you have any additional ideas!

—Reuven M. Lerner, reuven@lerner.co.il

Concerns about certification

In reading the recent articles in *LJ* about Linux certification I have become concerned. I read a lot about why certification is important to us all, but what I did not see is any disclosure of self interest. Will P. Tobin Maginnis or Sair, Inc. make any income from this certification process? Will Dan York or Linuxcare? Dan is a technical instructor. Will generating an environment where certification is required create additional billable training opportunities? If this is a money making venture, then be forthcoming about it, and don't veil it with the argument that this is what is good for the community. I'm not opposed to making money, but lets be honest about the objectives. (It can be profitable and good for the community, but if the primary objective is generating revenue the community will not be well served.) Anyone who thinks the primary motive of other proprietary software certifications (you know who they are) is to better serve the community, needs to follow the money to where the real motivation lies.

Another question this has raised is, can anyone offer a Linux certification? If so, can I offer the Geo's certification and administer tests and collect fees? This could lead to fragmentation and diminish the value of all Linux certifications. Which one is credible?

So what is the remedy? I don't know, but here are some thoughts. I think in order for a certification body to be credible it needs to:

- 1.) Be a non-profit organization. Only then will we know for sure that the goal is community, not money.
- 2.) Have all volunteer staff. No paid employees. Salaries to employees of even a non-profit organization can lead to self interest. Volunteers will do it because it is good for the community. Linux User Groups could be utilized to administer tests.

3.) Have no expenses outside of direct expense of test administration. No office, no expense accounts, etc.

4.) No fees outside of direct expense of test administration. Like printing, postage, website.

5.) Full public disclosure of all finances. The community needs to know that the interest of the certification body is only the community.

This may seem extreme but I think it is necessary in order to keep the objectives of the testing body in line with the community. Its true that other software certification bodies don't work this way, but again, follow the money to understand their motivation. I am not opposed to making money. I am a working programmer. I think there are plenty of opportunities for organizations other than the testing body to charge fees for training and support, but letting them create a need for training via creating a need for certification that they will administer and profit from does not put the best interest of the community first.

I don't think I am alone. I think there is a (so far) silent majority that is concerned and is watching the certification issue unfold.

Any volunteers?

George Saich, gsaich@ibm.net

Nasty bug LJ #61 Pthread code

Hopefully you can redirect this letter

There are a real nasty bug in the code published in the Pthread artikle.

Puts a new /focus/ your online artikle on buffer-overflow.

Change:

```
--8<-----  
int main()  
{  
    pthread_t *tide; //  
    int i;  
    for(i=0; i<10; i++) {  
--8<-----
```

TO:

```
--8<-----  
int main()  
{  
    pthread_t tide[10];  
    int i;
```

```
for(i=0; i<10; i++) {  
--8<-----
```

—Klaus Pedersen, klaus.pedersen@nokia.com

about Linux's standards article

Title: LSB: more facts, please

I'm dissapointed about Daniel Quinlan's article when it talks about Linux Standards Base (LSB) because he doesn't explained enough the current state of this standard, nor if there are concise LSB plans or scheduling, and if there are'nt, why not?. What's happening in the LSB efforts?. I believe is a good idea to keep comunity informed in that way.

Also I'm dissapointed on all distributions because I think LSB must be one of the highests priorities, LSB should be there some time ago and even it seems that currently LSB is in it's early stages.

Reading opinions from distributions' people it's also not very gratifying, even they say don't want to see fragmentation they actually bundle generic products as distribution specific. This causes more confusion to the market. Distributions should enforce themselves to avoid this in order to make Linux a stable and a really homogeneous Open Source platform.

IMHO Linux is just starting to compete in the marketplace and distribution competition should not rely on things that produce fragmentation. The LSB is a must and it needs much more than promises and good wishes.

I hope actions will be taken fast to avoid this in the future.

Thanks for this media to provide user feedback

—Ulisses Alonso Camaro, ualonso@slabii.eleinf.uv.es

Re: Linux and E-Commerce article in LJ 63

To: Yermos Lamers

On page 41 of the July 1999 issue (number 63) of the *Linux Journal*, you state in your second bulleted point on that page (sixth bullet in 'Putting it All Together'):

If the transaction was authorized ...
[...]b4 The reasoning is that if someone steals your credit card, it is unlikely they have your address as well.

Say what??? That's for the public to see, right?

The chance of having the credit card loss occur as a result of loss of a purse or wallet is significant. In every case that I'm aware of, the driver's

license contains both address and zip/postal code. Please check the contents of all of your employee's wallets, and see. Then note if some other unique number that is also known to the credit card company is present. I don't know for fact's, but think that the home phone number would be a better check. Here in the states, it is possible to locate a super telephone directory on line, to see if the customer's name matches a phone number, which is much less likely to show up in the wallet. Even so, even this is not fool resistant, much less fool proof. If the baddies know that is part of the check, they'll consult the super phone book as well, before placing an order with you.

No, this is not the 'secure' way of authenticating the customer. I know that 1-2 percent is industry average, but there are ways to improve that. Making certain that the billing address is the same as the shipping address will help. This may have been what you implied with your address check, but if so, it may have be to subtile, and at the same time, not subtile enough.

Otherwise, a good article. I like the 'real world' approach to solving your proplems.

—Bill Staehle, staehle@netvalue.net

False report

On page 18, paragraph 3, line 2 of the June issue, you say that ssh fails to compile with glibc-2.1. I have been using glibc-2.1 for about 4 months now, and never failed to compile ssh. BUT, you have to have a very clean and neat set of include files in the include path, in order for the job to be accomplished. Mixing headers from new and old versions of glibc doesn't help a bit.

Thank you for your time and BTW, thank you for your magazine.

—Gerasimos Melissaratos, gmelis@eexi.gr

Netscape and glibc 2.1

Thanks for a great magazine!

I read an article in the June99 issue on p18 by David A. Bandel about Glibc 2.1 vs Glibc 2.0 and problems with Netscape. I don't know anything about his problem except what he wrote, but I had some problems with Netscape and Stampede (based on glibc 2.1). At first I blamed Glibc 2.1 but the problem wasn't there, it was in libstdc++, Netscape 4.51 didn't like the libstdc++.2.7.x that came with Stampede but after putting just the binaries from RedHat 5.2 in /usr/local/lib it worked fine. Netscape 4.6 needs libstdc++.2.8.x and the RedHat version seems to work fine.

BTW: I think I read somewhere on the internet that there had been some sort of mixup with the Netscape binaries so that they put the libc5 and

the glibc 2.0 in the wrong places. Maybe that's why he had to change to the libc5 version.

BTW2: I can recommend a great program for filemanagement, it's called FileRunner and it's able to do a lot of things, take a look at: <http://www.cd.chalmers.se/~hch/filerunner.html>

—Chris, chris.bele@2.sbbs.se

Re: Question on an article

Scott,

Every package I review in *LJ* has been compiled and run on a Linux system. Just off hand, I can't tell you which system, but it will be either a modified Debian 2.1 (w/ a 2.2 kernel) or a Caldera OpenLinux 2.2 system (with updated libraries).

The biggest problem folks face in compiling things on Linux are libraries. You need lots of them and current ones.

As for compiling Saint: I looked at the Saint site. It specifically mentions both Red Hat 6.0 and Caldera OpenLinux 2.2 as capable of compiling Saint (there were problems originally with glibc-2.1).

I don't create RPMs or Deb packages for myself, so I won't start doing it for others. I just don't have the time or disk space. If you are having problems compiling the program, send me the error messages, and perhaps I can help decipher them.

Besides, if you're running a system with libc5 or glibc-2.0.7, any binaries I send you will just segfault.

I'd be happy to help you get Saint compiled. Just make sure you have current copies of libraries I listed, get the latest version of Saint, and if you have problems, e-mail me the error messages, and I'll see what we can do.

Good Luck,

—David A. Bandel, dbandel@ix.netcom.com

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

[Advanced search](#)

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

[Advanced search](#)

upFRONT

Various

Issue #64, August 1999

LJ Index, Earth-Shaking Harbingers and more.

Thoughtful Wishing

Make it happen, or bet on when it will

iTux: I envision a Linux-based iMac. Call it an iTux (too bad, the domain is already taken). Dress it up in those handsome penguin colors (or lack of them), rather than the iMac's silly "flavors". Use nothing but end-of-life components with no fan and no floppy so it can be built dirt cheap. Boot from the CD-ROM drive. Solder everything to the boards including memory, with the possible exception of a communications slot. Now we're talking about a black & white box: simple and reliable as an old phone. Shoot for a \$500+ price point, complete with all network connections, OS and software including management agents. Put the money in memory and display, which ought to be active matrix, if there's any way. Build two classes: corporate and home. Skew cute for the home version and add a slot for a Sony Playstation module. Message: for the price of a Playstation, get a computer too.

Internet Floppies, Any Size: FTP and e-mail killed the floppy. Now it is time to unburden those (social as well as technical) protocols. Let's create Internet floppies. Create easily understood ways to create, share and manage virtual Internet file servers of 10 to 100MB (or larger). This will eliminate the need for virtual private networks, e-mail attachments and complicated remote access routines. Any user should be able to sign up for one with his or her ISP, then decide who else can use it. Versions of this already exist at dedicated sites, such as NetFloppy, <http://www.NetFloppy.com/>. Why not at any site, any ISP, any business, using free and open software?

ISPs would get stickier sites and happier customers. Businesses would get turn-key field file-service solutions for maybe \$2 per user, instead of the \$50-75 it

might cost from a file-service vendor. End users would get easy ways to pass files around and back up hunks of their local directories.

The Numbers

Alexa Internet is best known for three things:

1. A neat little browser accessory that reveals stats, domain ownership and other non-obvious information. Alexa gathered pages for 220 million unique URLs and found that only 6 percent of Web content has changed in the last three months.
2. Regularly backing up the entire contents of the Web. “Links in” are determined from their ongoing web crawls.
3. Selling to Amazon.com for something like \$250 million.

The Alexa browser accessory does not yet work on Linux, but that didn't stop us from using it to tally up some comparative numbers for various Linux-related web sites. The results are below—we'll leave the interpretations up to you.

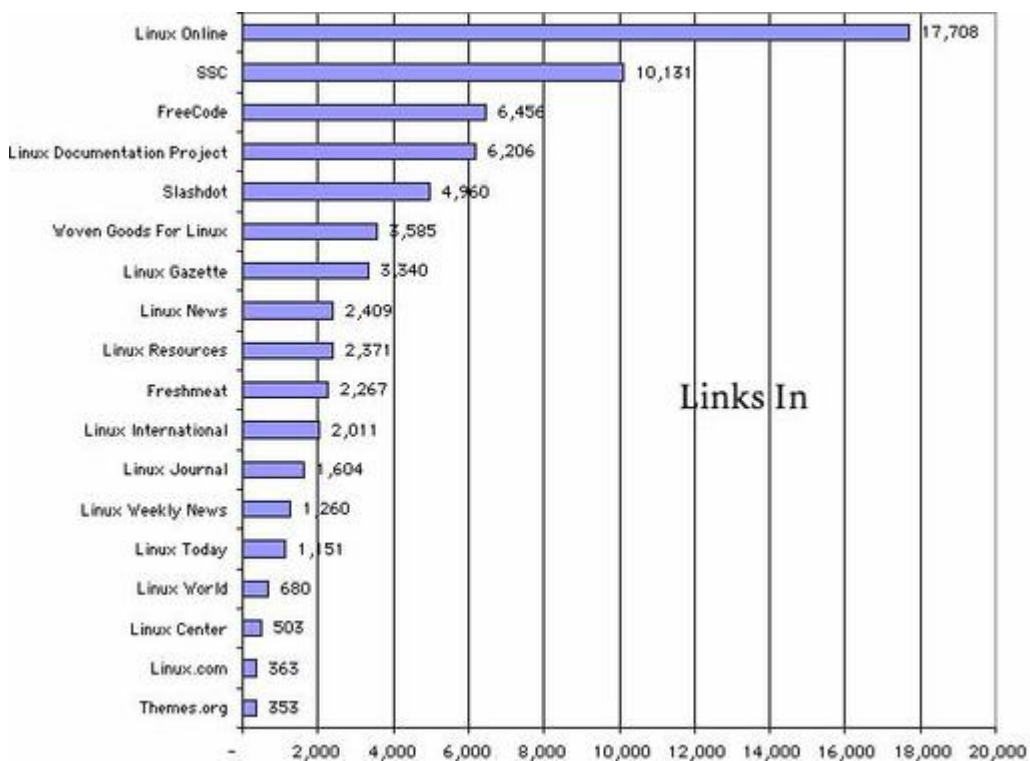


Figure 1. Linux Resource Sites—Links In

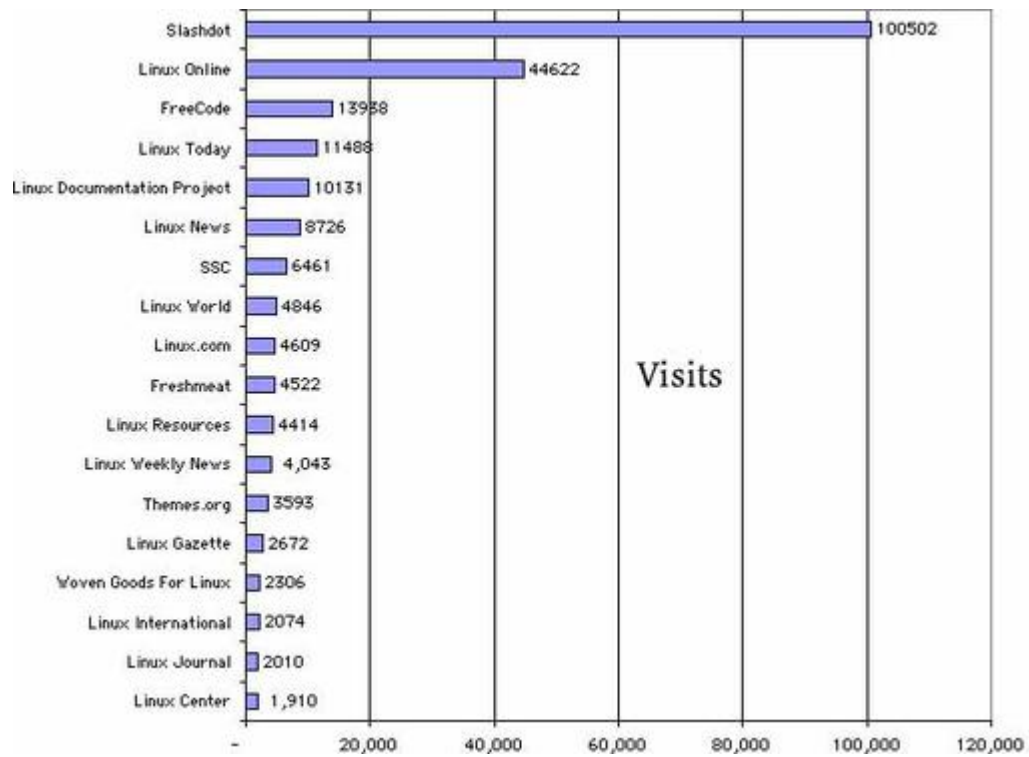


Figure 2. Linux Resource Sites—Visits

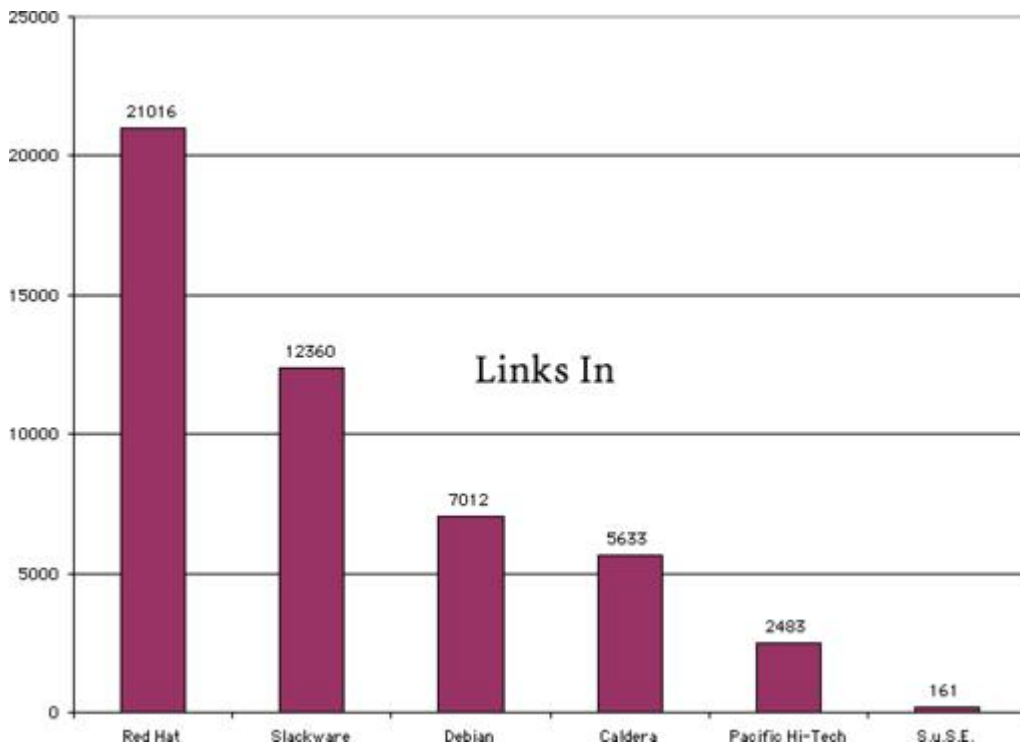


Figure 3. Linux Distribution Sites—Links In

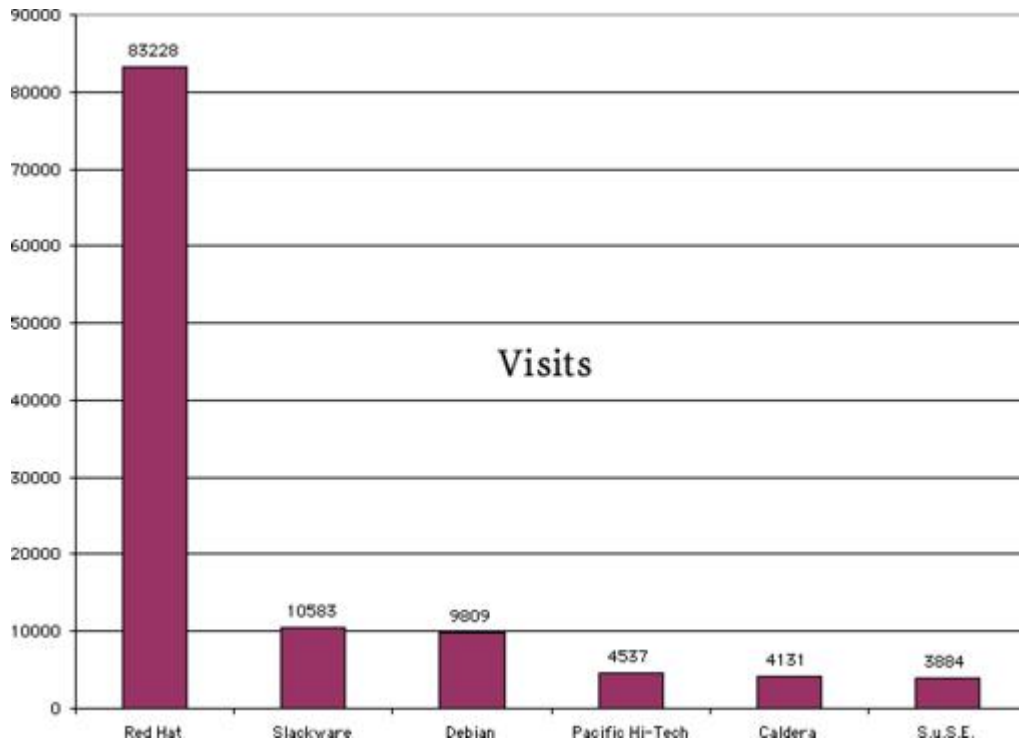


Figure 4. Linux Distribution Sites—Visits

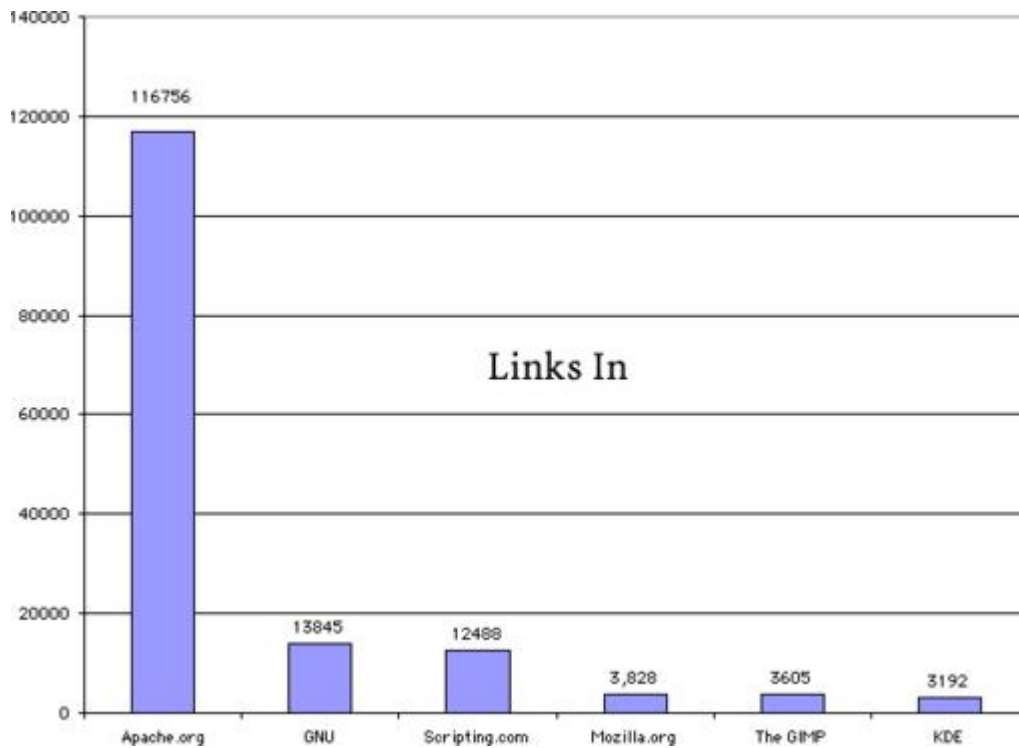


Figure 5. Linux Development Communities—Links In

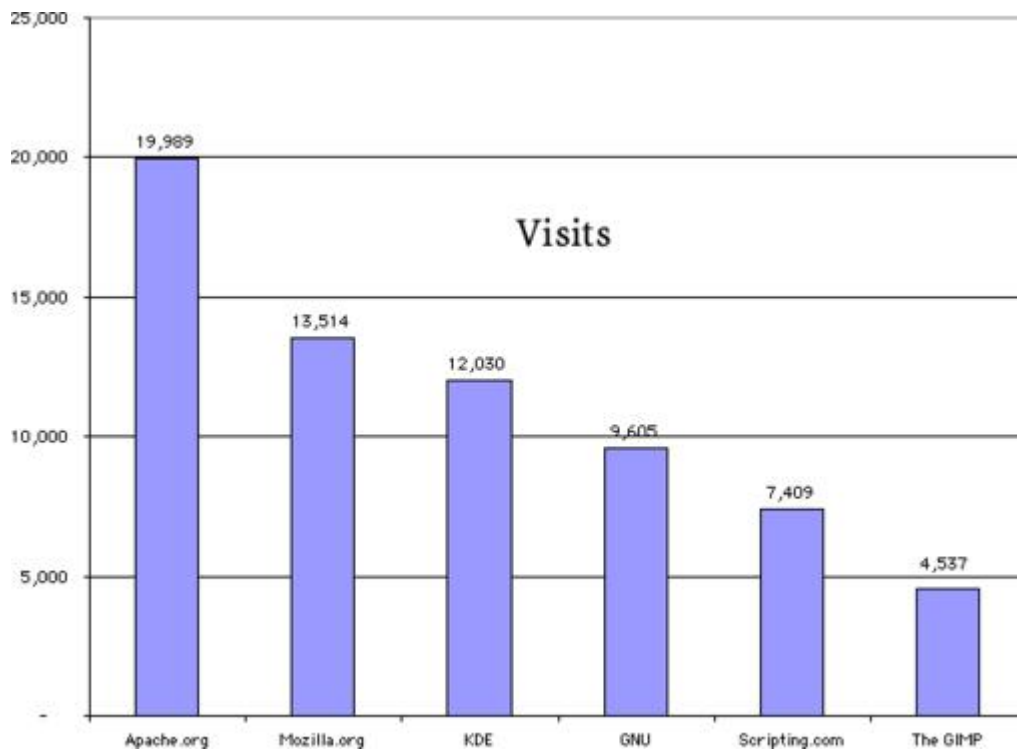


Figure 6. Linux Development Communities—Visits

Notes: These numbers were collected on 5/13/99. “Links in” are from surveys of pages served across the entire Web. “Vists” are accumulated over the previous six months of activity by Alexa users, nearly all of which are on Windows clients. Some sites, such as Scripting.com and Slackware, contain high percentages of links and visits that are not Linux-related. One major site, Linux.com, went live from VA Linux Systems only in the last few days of the survey period. A follow-up survey on 6/1 showed a 24.6 percent increase in visits to that site.

LJ Index—August, 1999

- Web pages containing the word “the”: 130,310,754
- Web pages containing the word “a”: 118,829,051
- Web pages containing the word “Microsoft”: 10,469,074
- Web pages containing the word “Linux”: 628,828
- Web pages containing the word “competitors”: 612,246
- Web pages containing both “Microsoft” and “competitors”: 53,368
- Web pages containing both “Bill Gates” and “our competitors”: 371
- Web pages containing both “Linus Torvalds” and “world domination”: 239
- Penguin species: 17
- Penguin species native to areas north of the Galapagos Islands: 0
- Number of web sites with the word “Antartica”: 5,080
- Number of web sites with the word “Antarctica”: 65,180
- Number of web pages with the word “penguin”: 435,110

- Number of web pages with the words “penguin” and “antarctica”: 3,949
- Number of web pages with the words “penguin” and “Linux”: 14,819
- Number of web pages with the name “Linux Thorvalds”: 10
- Number of web sites with the name “Linus Thorvald”: 27
- Number of web sites with the name “Linux Torvalds”: 940
- Number of web sites with the name “Linus Torvalds”: 17,580
- Journalists among Linus' parents: 2
- Maximum seating for the room Comdex Spring booked for Linus Torvalds' keynote: 75
- Maximum seating for the room where Linus ended up giving his Spring keynote: 850
- Estimated number of people who attended Linus' Comdex Spring keynote: 1,200
- Pages Hotbot finds with “Anonymous Coward”: 81

Sources

Pete & Barb's Penguin Pages

1. AltaVista, 5/11/99
2. Hotbot, 5/21/99
3. Linus Torvalds, speaking at Spring 99 LinuxWorld
4. Maximum occupancy listings posted on the walls of those rooms, plus observations by *Linux Journal* personnel in attendance.

Earth-Shaking Harbingers

There are now Linux productivity applications that are sufficiently attractive that I am using them. This means I am finally learning to use a mouse. I am actually using GnomeCard to maintain my phone lists now. While regular users are learning Linux command lines, I'm learning to point and click. What makes GnomeCard interesting is that it has the characteristics of a good point-and-click Roll-O-Dex, but on the backend it uses a straight flat-text Vcard format. This means you're not locked in—you can use it with other tools. —Eric S. Raymond to Doc Searls, May 1999

Vendor News from Linux Expo

FairCom has a new release of their c-tree Plus file handler, which includes the Crystal Reports Driver to make ex-Windows users feel comfortable. File encryption is available using algorithms that can be decrypted only by FairCom's servers. The server includes an SDK (software development kit) to allow developers to customize the encryption process, resulting in an increased

level of security. When talking to Winston Atkinson of FairCom, I learned that System Development Group, Inc. (SDG) is using FairCom's products for **XLN**, their Enterprise Management Software. XLN provides integrated system modules for a full range of functions necessary for planning and scheduling of work, such as production, manufacturing, distribution, data collection and accounting. XLN supports Linux and is Y2K-compliant.

Metrowerks is the manufacturer of CodeWarrior, a developer tool package that now supports Red Hat Linux. I talked to Jean Bélanger, Chairman and CEO, about why they chose to port to Linux, and he told me management had initially turned down project requests for the port. He said Metrowerks engineers did the port on their own and presented it to management as an accomplished fact. Metrowerks is so happy with this effort, they intend to port to other Linux distributions in the near future. Most of the work had already been done by the time they ported to Solaris over a year ago, but it was still not easy, as CodeWarrior uses many system-level services. Mr. Bélanger said Metrowerks hopes to become the de facto standard for developer tools on Linux.

SGI has just released their file system, XFS, to the Open Source community. Dave McAllister told me SGI is helping to fill the gaps in Linux technology by making products, such as XFS, GLX and OpenVault, Open Source. He feels that Linux fits nicely into their high-end product offerings and is a good match for clients who want innovative systems.

Rebel.com is the new name of Hardware Canada Computing, the company that bought the NetWinder from Corel Computing. They were at Linux Expo showing off the small and powerful NetWinder, with a new rack mount that holds two of the little servers. I knew the NetWinder was small, but didn't realize just how small until confronted with one—it's not much bigger than a notebook computer.

Zenguin is a new start-up company put together by Scott McNeil, Bodo Bauer and Eureka Endo, all formerly with SuSE. Zenguin is creating an installer for Linux applications to enable point-and-click installation of future Linux tools. The installer will include a knowledge database specific to the needs of the ISV's software and the variables of the Linux system. In other words, this product will act as an install-shield for Linux. This is something Linux has been needing—kudos to these guys for seeing the need and stepping up to bat.

Digital Creations introduced their Zope Portal Toolkit. I saw a demonstration and was quite impressed with its power and ease of use. It provides news, search, directory and membership services. Later, I was talking to Dan York

about this product and discovered he is already using it, so I got him to agree to write a review for us.

3R Soft Co., Inc. has released their web-based e-mail server program, MailStudio 2000. Talk about ease of use—they advertise it as easy to install, easy to use, easy to manage and easy to customize. It is a slick application that supports multiple languages—a powerful mail engine.

AbiSource has shipped the preview release of AbiWord 0.7. We have a review in this issue by Craig Knudsen. I like their motto, “Show Me the Source!”, and their goal of open-source applications for the desktop.

X.org is now the steward of the X Window System. Their current release is X11R5.4 and is available for download at <http://www.X.Org/>.

Applied Information Systems, Inc. announced a partnership with Business Logic Corp. to provide applications for the Linux retail market. Their first product is the Standard Edition of the XESS Spreadsheet for the desktop. This impressive product is completely compatible with Excel and can even share files with it via Samba. I talked to Arthur Coston, President of AIS, who told me their first port to Linux was done five years ago by Michael Johnson—five years of support for Linux is a laudable accomplishment.

BEA announced the availability of BEA Tuxedo and BEA WebLogic Server for Linux running on Intel servers. These products provide enterprise middleware and Java application servers, allowing developers to write applications once, then deploy them without modification to any of the more than 50 platforms supported by BEA.

Collective Technologies, an IT consulting company based in Austin, Texas, is now providing Linux consulting services. They had the best t-shirt at the show, featuring a large picture of Einstein created by tiling multiple small copies of the same image.

JDH Technologies is now supporting Linux with its Web4M groupware solution that supports e-mail, news, phone, browsable library, slide show, audio conferencing and much more. All you need is a web browser and a network connection. See the “New Products” column.

Cyber Station of Canada is making Linux easy by providing products such as EZ Linux Command Card, EZ Linux Mouse Pad (containing the same information as the card) and EZ Linux Software.

International GNOME Support is a company formed to provide development and customization services for GNOME. The company will adapt GNOME to the needs of its clients, enabling it to be deployed in mission-critical settings.

TUCOWS has been acquired by B. Steinmetz Technology Holding International. TUCOWS (The Ultimate Collection of Winsock Software) is now operating as TUCOWS.com Inc. and is one of the largest Internet distribution sites featuring Windows, Macintosh, Linux and PDA software. Find them at <http://www.tucows.com/>.

—Marjorie Richardson

Resources

Strictly On-Line

Extending the Bash Prompt by Giles Orr demonstrates the methods of customizing your xterm prompt in a Bash shell. Standard escape sequences can be incorporated to include user name, current working directory, time and more. Additionally, Mr. Orr shows us how to change the color of the prompt and manipulate the xterm title bar.

A High Availability Clustering Solution by Phil Lewis is the story of building a high-availability solution for his company, Electec. He reviews several clustering solutions, discussing cost and downtime for implementation. Also covered are load balancing, node takeover, networking hardware, partitioning and resynchronization. This article provides a thorough look at clustering and gives administrators a good start in building their own cluster.

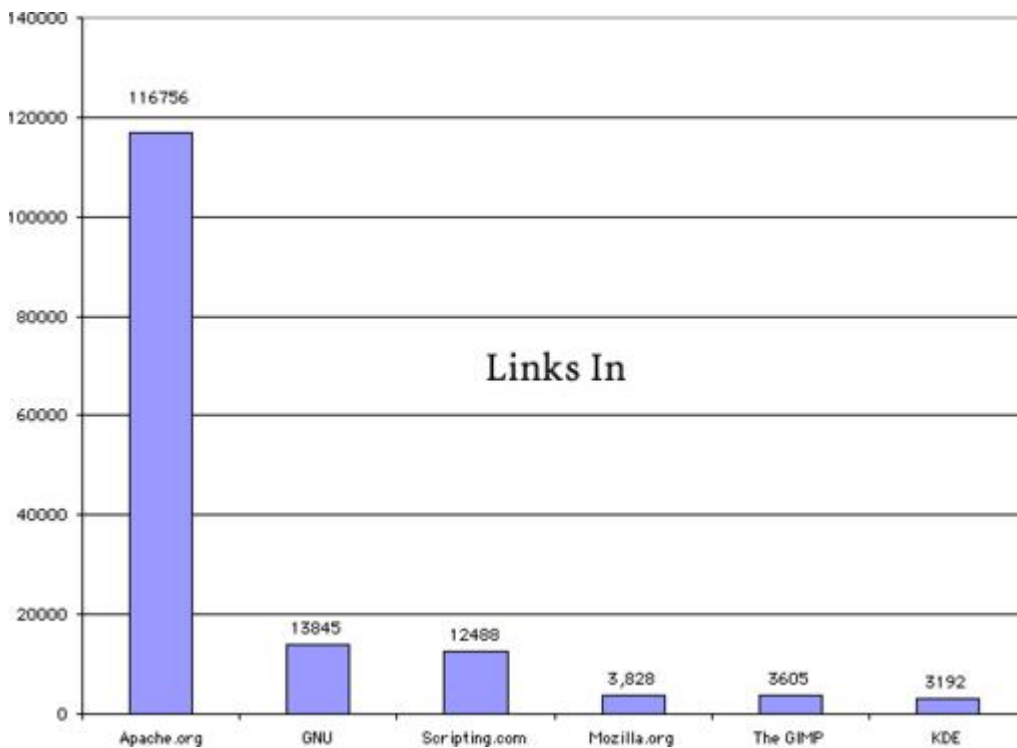
Introduction to Sybase, Part 3 by Jay Sissom tells us how to write a web application using Sybase. Mr. Sissom begins with a discussion of some SQL Server basics, such as transactions, logs, backups and database consistency. He ends with a full-blown web application for providing an on-line bookstore. All code for the example is available on our FTP site.

Plug and Play Hardware under Linux by David Cantrell is a discussion of PnP sound cards—how they work and how you get them to play nice on your Linux box. Installation, configuration, usage and even compiling the kernel is all here for your edification.

Open Source Remote Sensing Effort by Dr. Shawana P. Johnson tells us all about a project of developing remote-sensing software for the Open Source community. The goal of this project is to bring space to your doorstep.

A book review of "Linux: The Complete Reference, Second Edition" is presented by Ben Crowder. Find out what's in the book and if it could be useful to you.

OS Sucks-Rules-0-Meter



This operating system quality and approval metric is based on a periodic AltaVista search for each of several operating systems, directly followed by "sucks", "rules" or "rocks". It can be found on the Web at electriclichen.com/linux/srom.html--thanks to Don Marti.

Stop the Presses

Jon "maddog" Hall will be leaving Compaq on June 18 to join the team at VA Linux Systems. He struck the deal with Larry Augustin of VA Linux Systems during Spring Comdex. I talked to him on June 8 while at the USENIX Conference in Monterey, CA. Jon is actually a member of the USENIX board. The big question I asked was why he is leaving Compaq. He gave me several answers. The bottom line is because VA is a Linux company. If VA gives him a computer for his home, that computer will be running Linux, so it will be able to talk to the firewall and everything and his documents won't be in Microsoft Word format.

Jon has worked for Digital Equipment Corporation (DEC) for 16 years; DEC was bought by Compaq last year. At DEC, he worked as an engineer, a marketing manager, a product manager and a Linux evangelist. He went to DEC to help them create the best UNIX system available, and he feels they have done that. Other people agree. He also introduced DEC to Linux, and they are now selling Linux systems.

Compaq has been selling servers without licenses for years, and today many of those are Linux systems. One problem with Compaq/DEC was a type of culture clash: whereas DEC sold thousands of systems, Compaq sold millions. Thus, the sales techniques were different all the way down to what could be given away at a trade show. Also, his local boss has to approve him traveling to Texas to talk to management; in other words, it's a big company with many employees and all the red tape that goes with being big. Jon will find a much more relaxed easier to deal with.

Working for VA will give Jon the opportunity to do the two things he wants to do on a full-time basis: evangelize Linux and put more time and effort into making Linux International a bigger and better organization. That is, he wants to create a real board of directors for Linux International, get the charter finished and get every company possible involved in it. One way for Linux International to do that is by initiating projects to which both big and small companies can contribute. Basically, get everyone involved in fun things, e.g., workshops. In other words, grow Linux International and give it a higher degree of visibility.

Larry has basically agreed that Jon's job will be evangelizing Linux and fixing Linux International. For the moment, at least, Jon will continue to live in New Hampshire; VA is based in Silicon Valley.

—Phil Hughes

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

[Advanced search](#)

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

Advanced search

New Products

Ellen Dahl

Issue #64, August 1999

TeleUSE Personal Edition, /BriefCase 3.0 SCM Toolkit and more.

TeleUSE Personal Edition



Aonix announced that TeleUSE version 3.2 for Red Hat Linux is now available in a full-function Personal Edition. TeleUSE is a comprehensive tool for designing and building Motif-based GUIs in C, C++ or HTML on Linux systems. Features include Motif 2.1 support, cross-platform capability with TeleUSE QuickPort, and widget libraries. Personal Edition licenses are \$199 US per copy with standard edition licenses for enterprise use at \$3,950 US. Contact Aonix for international pricing.

Contact: Aonix, 595 Market Street, 12th Floor, San Francisco, CA 94105, Phone: 800-97-AONIX, E-mail: info@aonix.com, URL: <http://www.aonix.com/>.

/BriefCase 3.0 SCM Toolkit

Applied Computer Sciences, Inc. announced that /BriefCase Release 3.0, their advanced, enterprise Software Configuration Management solution for UNIX/Linux software development, is available for free download from their web site. Release 3 implements an easy-to-use client/server interface to an RCS-based

central repository. Enhancements in release 3 include project work tree "replicas", cross-replica and cross-client-host lock integrity, private tags, command behavior options, product life-cycle support and more. /BriefCase Release 3 is distributed in source form under the GPL.

Contact: Applied Computer Sciences, Inc., 34522 N. Scottsdale Road #458, Scottsdale, AZ 85262, Phone: 602-465-0944, Fax: 602-465-1078, E-mail: BriefCase@applied-cs-inc.com, URL: <http://www.applied-cs-inc.com/>.

Linux-Compliant ISDN Cards

Harmonix Limited announced the availability of a range of Linux-compliant ISDN cards. The multiple Basic Rate and Primary Rate ISDN cards from Harmonix provide both the CAPI 2.0 and the Linux-specific Isdn4Linux software interfaces. Available with channel bundling, the cards are suited to any application using voice or data. All Harmonix cards are available with on-card protocols such as Bonding Mode 0 and 1, and an MLPPP-based channel aggregation that can easily saturate all 30 channels of a Primary ISDN rate circuit. The cards also support MVIP (Multi-Vendor Integration Protocol). Contact Harmonix for pricing and more information.

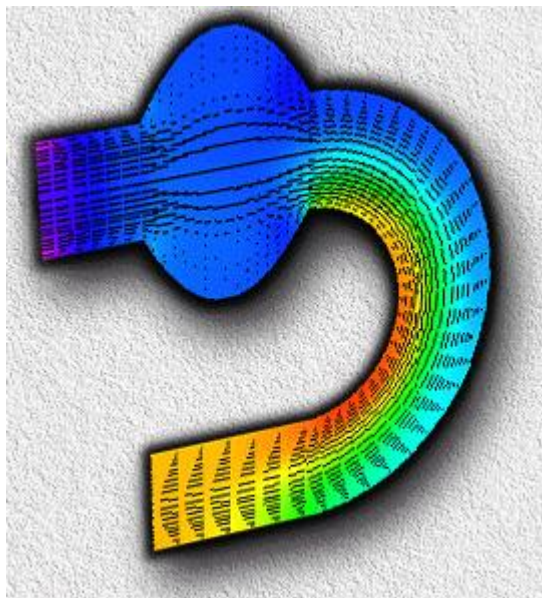
Contact: Harmonix Limited, Fry House, Sandhill Park, Bishops Lydeard, Taunton TA4 3DE, United Kingdom, Phone: +44-0-1823-433-711, Fax: +44-0-1823-433-722, E-mail: sales@harmonix.co.uk, URL: www.harmonix.co.uk.

Insure++ 5.0

Parasoft announced the release of Insure++ 5.0, an automatic error detection tool for C and C++. It runs on Linux and works at source code level to improve software quality, reduce development costs and accelerate time to market. Insure++ uses patented Source Code Instrumentation and Mutation Testing technologies to locate bugs in source code automatically. Operating in place of a compiler, it builds a database of all program elements for use at runtime to verify memory references and program implementation. The lite version is available free on the Red Hat Linux 6.0 CD. The full version of Insure++ includes two optional add-on modules: Inuse, the "Malloc Monitor", and TCA (total coverage analysis).

Contact: ParaSoft Corporation, 2031 S. Myrtle Ave., Monrovia, CA 91016, Phone: 626-305-0041, Fax: 626-305-3036, E-mail: info@parasoft.com, URL: <http://www.parasoft.com/>.

Diffpack 3.0



Numerical Objects announced a new release of Diffpack, a comprehensive development framework for multi-physics simulation. The object-oriented system is available on most UNIX platforms including Linux. New in Diffpack 3.0 is high-level support for finite difference methods and mixed finite element methods. Contact Numerical Objects for more information on licensing (university or commercial) and for pricing. A free Mini-Distribution or demo CD-ROM may be requested via the web site.

Contact: Numerical Objects AS, Forskningsveien 1, P.O. Box 124 Blindern, N-0314 Oslo, Norway, Phone: +47 22 06 73 00, Fax: +47 22 06 73 50, E-mail: sales@nobjects.com, URL: <http://www.nobjects.com/>.

PoPToP

Moreton Bay announced a pre-release of PoPToP (v.0.7.37), the PPTP server solution for Linux. PoPToP allows Linux servers to function seamlessly in the PPTP VPN environment, enabling administrators to leverage the considerable benefits of both Microsoft and Linux. The current pre-release version supports both Windows NT and Linux PPTP clients. On release, PoPToP will be compliant with the IETF PPTP Internet Draft. The PoPToP server is free software, licensed under the terms of the GPL.

Contact: Moreton Bay (USA), PO Box 51182, Pacific Grove, CA 93950, Phone: 831-656-9100, Fax: 831-656-9200, E-mail: sales@moretonbay.com, URL: <http://www.moretonbay.com/>.

Web-4M 2.5

Web-4M 2.5 from JDH Technologies now provides a comprehensive collaboration/groupware environment for Linux. The scalable, distributed tool suite supports e-mail, news, phone, the Browsable Document Library, the Interactive Slide Show, audio conferencing, chat, a white board, calendar, scheduler and more. It runs in conjunction with the Apache web server. Sample prices are \$1500 for a 25-user Web-4M server in business or government, \$1000 for 125 university users, and \$1050 for 175 K-12 users. An audio server module is available.

Contact: JDH Technologies LLC, 11834 Canon Blvd., Suite J3, Newport News, VA 23606, Phone: 757-873-4747, Fax: 757-873-8484, E-mail: info@jdhtech.com, URL: <http://www.jdhtech.com/>.

VSI-FAX Gold

V-Systems, Inc. announced their new fax solution for Linux, VSI-FAX Gold. The VSI-FAX Gold Series is a powerful, cross-platform, scalable client/server fax solution for UNIX, Linux and Microsoft Windows NT servers. The TCP/IP-based server uses a robust central fax engine that runs on a network host, providing reliable performance for all fax-enabled applications and desktop fax users. With its unified inbox/outbox capability, one can use an e-mail client to send and receive faxes. Please contact VSI for a product literature kit, an evaluation CD-ROM or to download software.

Contact: V-Systems, Inc. (VSI), 32232 Paseo Adelanto, Suite #100, San Juan Capistrano, CA 92675, Phone: 800-556-4874, Fax: 949-489-2058, E-mail: sales@vsi.com, URL: <http://www.vsi.com/>.

SourceOffSite Professional Edition v2.0



SourceOffSite, Inc. announced the 2.0 release of SourceOffSite Professional Edition. The product is a high-performance tool that enables distributed

development teams to share Microsoft Visual SourceSafe databases (sourceoffsite.com/product.phtml). Source*OffSite* offers remote access and cross-platform functionality among UNIX/Linux clients and Windows NT servers. The product is specifically designed for companies that need fast and secure access to a centralized SourceSafe database from any Internet connection. Price is \$149 US for one to nine licenses, reduced for ten or more licenses.

Contact: Source*OffSite*, Inc., 3200 Farber Drive, Champaign, IL 61822, Phone: 217-356-3213, Fax: 217-356-0135, E-mail: sales@sourceoffsite.com, URL: <http://www.sourceoffsite.com/>

ScriptEase v4.10c

Nombas, Inc. announced ScriptEase:ISDK version 4.10c, a multi-threaded, multitasking JavaScript/ECMAScript that allows one to extend the functions of products from mainframes to Linux, UNIX to PCs, MACs and embedded devices. ScriptEase enables developers to embed a fully functional, JavaScript-compliant, scripting language into applications, embedded systems and proprietary browsers. Pricing will vary; contact Nombas for more information.

Contact: Nombas, Inc., 64 Salem Street, Medford, MA 01890-1709, Phone: 888-766-6227, Fax: 781-391-3842, E-mail: nombas@nombas.com, URL: <http://www.nombas.com/>.

UltraSPARC Server for ISPs



EIS Computers introduced a complete set of products and support services targeted to meet the computing needs of Internet Service Providers. Central to this new product set, EIS introduced a new low-profile rack-mount UltraSPARC

server, the Fusion-iX/2, and announced a partnership with national system support provider Terix Computer, Inc. The Fusion-iX/2 includes dual UltraWide SCSI channels, 100baseT Ethernet, up to 1GB RAM, Sun's 360MHz UltraSPARC II(i) CPU, up to four hard drives, floppy, tape or CD-ROM, two PCI cards, a redundant hot swappable power supply and a rack-mountable enclosure. Pricing can be found at the web site.

Contact: EIS Computers, Inc., 207 W. Los Angeles Ave, #303, Moorpark, CA 93201, Phone: 800-351-4608, Fax: 805-383-1470, E-mail: info@eis.com, URL: <http://www.eis.com/>.

Server-In-a-Box

EMAC, Inc. announced The Server-in-a-Box (SIB) which provides control and monitoring of simple serial devices and complex systems over the Internet or a LAN. Users can interact with a standard browser from anywhere in the world. The SIB is based on a 486, 133Mhz single-board computer, and can run on 8MB of Flash and 8MB of RAM. The ability to boot and run from Flash eliminates the need for a hard drive, and the 486 processor does not require a fan. The SIB uses a modified Linux kernel, providing excellent security, and is fully compliant with HTTP v.1.1. Cost is \$895 US, and \$199 US for an optional dial-up modem.

Contact: EMAC, Inc., 11 Emac Way, Carbondale, IL 62901, Phone: 618-529-4525, Fax: 618-457-0100, E-mail: info@emacinc.com, URL: <http://www.emacinc.com/>.

TurboLinux 3.0.2-C, Chinese Version

Pacific HiTech announced the availability of TurboLinux in Chinese. Pacific HiTech will be providing investment and technical support to establish the Linux Research Center at Qinghua in Beijing, China's top science university. The center will support the CERNET data network. Pacific HiTech is also partnering with two of China's leading computer companies to establish a Linux call support center. TurboLinux Chinese has a list price of \$49.95. A free download version is available at <ftp://ftp.pht.com/> and at more than ten FTP sites in China.

Contact: Pacific HiTech, Inc., 586 East 9620 South, Sandy, UT 84070, Phone: 801-501-0866, Fax: 801-501-0889, E-mail: info@pht.com, URL: <http://www.turbolinux.com/>, www.pht.co.jp.

XA Series



IndyBox Systems announced an offer of AMD K6-2 350MHz-based Linux workstations starting at \$425 US, with one of the many Linux distributions pre-installed, including Debian, Red Hat, SuSE, Caldera, Mandrake and Slackware. The boxes are available in a mid-tower or 4U rack configuration. IndyBox also introduced a complete line of rack-mountable AMD and Intel-based Linux systems starting at \$625 US.

Contact: Indybox Systems, Inc., P.O. Box 3564, Carmel, IN 46082-3564, Phone: 317-846-9762, Fax: 317-870-1823, E-mail: sales@indybox.com, URL: <http://www.indybox.com/>.

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

[Advanced search](#)

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

[Advanced search](#)

Best of Technical Support

Various

Issue #64, August 1999

Our experts answer your technical questions.

Problem Upgrading the Kernel

I'm trying to upgrade to kernel 2.2.5. When I type **mkinitrd /boot/initrd-2.2.5.img 2.2.5**, the system returns:

```
mount: the kernel does not recognize /dev/loop0 as a block device
Can't get a loopback device
```

What should I do? What has gone wrong? —Mun Hon Tham,
mhtham@cyberway.com.sg

For some reason, automatic module loading is broken (you could do **modprobe loop** to load the module by hand). I am willing to bet you installed all the updates from the Red Hat FTP site, including the new modutils from the 2.2 directory. Unfortunately, modutils has a completely broken kernel that causes the message you are seeing. I do not know why Red Hat hasn't taken that broken package off their FTP site. In the meantime, all you need to do is downgrade to the previous modutils by typing:

```
rpm -U --force modutils-version.i386.rpm
```

—Marc Merlin, marc@merlins.org

Cylinder Question

The Installation guide says the partition that contains /boot must be below cylinder 1023. My hard drive has 1400 cylinders. How do I find out which cylinders are not being used and if /boot will fit below cylinder 1023?

I will be using a dual-boot system with a DOS partition. My hard drive is 10.9GB and DOS/Windows is the primary partition, using 4.5GB. Can you help? —Jason Hipsher, jclive@netscape.net

The most important factor that slips up new users is LBA mode drives. Logical Block Addressing translates cylinders into heads, dividing one by two and multiplying the other by two, so the numbers still work but you have fewer cylinders.

The problem is that with a 10GB drive, you most likely do not have 1400 cylinders. You probably have far more, and LBA is enabled on your drive (this is done in your BIOS and is normally enabled by default). You cannot turn it off without reloading your DOS/Windows partition.

You may be able to use the “linear” addressing feature in LILO to get around this problem, but it doesn't always work. A neat trick I use to get around the issue is to create a small (8MB is usually more than enough) partition at the very start of my drive to store my kernels. You may need PartitionMagic or some other partition-manipulation tool to do this without loss of data, but it will give you a place to store files that is guaranteed to be lower than the 1024th cylinder. You can then mount this at boot time (I put it under /img), so you can store your kernels there. Linux itself can be anywhere on the drive. —Chad Robinson, chadr@brt.com

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. —Mario Bittencourt, mneto@argo.com.br

X Windows vs. X11

What, if any, is the difference between X11 and X Windows? It seems as though both terms can be used interchangeably, but at the same time, it seems as if they are different servers. Is X Windows a shell or server on top of X11, or are they entirely different? I am not talking about look and feel, but the basic meaning. —Chris, kender_kin@hotmail.com

The terms are often used interchangeably, which can be confusing. The “official” name for X is “The X Window System”. Some purists get irritated when people say “X Windows”, but it's usually safe to say “X”.

You are actually talking about the same program. What's different is that the first term (X11) refers to the specific version of X. You're actually missing part of it—an Rx is usually included after it. For example, X11R5 refers to a popular and still commonly seen release. X11R6 is what most new Linux distributions are

shipping with now (in the form of XFree86, an open-source distribution of X). — Chad Robinson, chadr@brt.com

Another X Question

When I type **startx**, the response I get is:

```
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
```

Now, this is kind of irritating—I have used XF86config, but it doesn't seem to have the driver for the SIS6215 (4MB RAM). We have had X up, but only after we made some adjustments—we could view it in only 16 colors. Since this is just the 2.0.36 kernel, is there a chance that I must upgrade it? Do I need to get the XFree3.3.3 version to get this thing alive? (The school's hardware is a P166, SIS6215 videocard, Digital screen.) Do I have to reconfigure everything, or maybe install the new kernel and XFree version? —Mathias Bakkejord, chenrazee@hotmail.com

The newer the X version you use, the better your chances of finding a driver designed for your card. Upgrading your Linux kernel won't help here, although there are other good reasons to do so.

However, you can probably get your card working in at least 256-color mode by using the SVGA driver. This is a good idea anyway, since it will give you a chance to learn what the clock rate and refresh settings for your card and monitor need to be. It would be a good idea to try this before trying to find a specific driver for your card. —Chad Robinson, chadr@brt.com

Changing the Login Prompt

I would like to change the pre-login banner on my Red Hat 5.1 system; however, each time I change /etc/issue, “the system” changes it back. How do I get around this? —Philip Lowe, uplowe@mcs.drexel.edu

The banner file /etc/issue is created by the script file /etc/rc.d/rc.local which runs whenever the machine is booted. In order to modify the issue file, you'll need to edit the relevant lines in rc.local; either comment out the lines that overwrite the current issue file, or incorporate your changes so the script generates the file you are after. The lines in question are documented with comments, so you should have no problem finding the lines to change. —Vince Waldon, Vince.Waldon@cnpl.enbridge.com

Problems Connecting

I am new to Linux. I have successfully installed Caldera OpenLinux 2.2 and have configured KDE, etc. I want to use **kppp** to dial up to the Internet. However, when I attempt to dial up, I get an error message indicating "Sorry, the modem is busy."

I have run LISA to install the modem on /dev/ttys1 (Com Port 2). I have tried setting kppp to /dev/modem/ and several other tty devices (when set to ttys0, I get a different message indicating the modem cannot be found). The modem is an internal US Robotics, 56K modem (not a winmodem). All attempts to access the modem (e.g., via Test Modem in the kppp setup) result in the same message.

I have referenced several texts, including the *kppp Handbook* and followed all the tips; but the modem still has the same problem. I have researched the Internet newsgroups where several other people have mentioned this issue, but no one seemed able to answer the question. —Shannon Brown, linux@rubicongulch.com

Make sure you have no lock file for your modem. Your first attempts to set it up may have accidentally terminated a process before it could remove its lock file. You normally find these in /var/spool/locks, depending on the program that creates the lock. You should probably use /dev/ttyS1 (note the upper case S) to access your modem if it is on COM2 (or /dev/ttyS0 for COM1). —Chad Robinson, chadr@brt.com

Enabling lockd

I don't know how to use PCNFSD to enable **lockd** and **share**. Red Hat technical support refuses to answer me. Where can I get documentation and how can I implement lockd? —Joan Cartigny, cartigny@bourelly.com

Last I checked, the tech support you get with a Red Hat box is installation tech support, not detailed configuration. If you need configuration tech support, you need to purchase a support contract with Red Hat, LinuxCare or another provider.

Are you sure you mean PCNFSD which is used only by some DOS/Windows clients? If you mean NFS, you need to use **knfsd** to get lockd support, and for this, you should use a 2.2.x kernel. You can upgrade yourself, or get Red Hat 6.0 which includes them by default. —Marc Merlin, marc@merlins.org

KDE and Slackware

I'm trying to install KDE. When I try to run it, it says I'm missing libstdc++2.9. I asked about this on a mailing list and was told that I was probably missing a C compile library. I then added every C compiler listed in disk d1 to my Slackware 3.6 installation. Still didn't have libstdc++2.9. On an official KDE site, it says I'm supposed to edit the .xinitrc file, but when I attempt to do that in **vi**, it says .xinitrc is not a regular file and then declines to edit it.

Any and all information about the Slackware 3.6 installation of KDE would be helpful. —Lisa Zuckerman, blueink@netzero.net

This is bad news because .xinitrc is a regular text file—perhaps yours is corrupted.

Slackware is missing many recent libraries and is still using libc5. If you grabbed KDE binaries, they are most likely compiled against glibc, and they won't work on your machine. libstdc++ is a C++ library that comes with glibc, I believe.

You can find and install all the libraries yourself, but I recommend switching distributions. The most recent ones have KDE built-in. Slackware is usually a year behind, although a more recent version of Slackware may help. —Marc Merlin, marc@merlins.org

Font Problems Across Platforms

I have MkLinux installed on my Macintosh notebook computer (PB 3400/200). When I am at work, I like to use my Sun (Ultra 2) to display Windows from my powerbook. To do this, I use telnet to get to my powerbook from the Sun:

```
setenv DISPLAY SunComputerName
```

At the Sun prompt, I type:

```
xhost + MacName
```

Now I can display xv, Netscape, xterms, Ghostview, etc. to my nice Sun video monitor. But I can't seem to get Applixware or LyX to do the same. Applixware complains that it doesn't know the font path (X-Server problem—unable to access fonts). LyX displays a window with a splash screen, but it dies as soon as I try to open a document or start a new one. —Vince D. Dupperron, vince@shellus.com

You are apparently missing fonts on the Sun machine. I believe Applixware has some documentation on how to run the application remotely or over NFS. The

idea is that you must install the required fonts on your Sun. LyX probably has the same problem. —Marc Merlin, marc@merlins.org

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

[Advanced search](#)

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

Extending the Bash Prompt

Giles Orr

Issue #64, August 1999

Terminal and xterm prompts can be created incorporating standard escape sequences to give user name, current working directory, time and more.

Descended from the Bourne shell, Bash (Bourne Again Shell) is a GNU product that is the standard command-line interface on most Linux machines. It excels at interactivity, supporting command-line editing, completion and recall. It also supports configurable prompts—most people realize this, but may not realize how useful it can be.

Most Linux systems have a default prompt in one color (gray) that includes your user name, the name of the machine you are working on and your current working directory. In addition, you can display even more information, use ANSI colors and manipulate the title bar of an xterm to provide useful information.

Beyond looking cool, prompts are also useful for keeping track of system information. One idea with appeal to many is the use of different color prompts on different machines. If you have several xterms open on different machines or if you tend to forget which machine you are working on, you'll find this a great reminder.

To change your prompt, you need a basic understanding of shell programming and UNIX utilities. The more you know, the more complex the prompts you will be able to create.

The appearance of the prompt is governed by the shell variable **PS1**. Command continuations are indicated by the **PS2** string, which can be modified in exactly the same way. Since controlling it is exactly the same, I'll mostly be modifying the **PS1** string. (**PS3** and **PS4** strings are also available, but are never seen by the average user. See the Bash man page if you're interested in their purpose.) To change the way the prompt appears, change the **PS1** variable. For experimentation purposes, the **PS1** string can be entered at the prompt to

show the results immediately. Doing so affects only your current session. If you want to make a permanent change, modify the `~/.bashrc` file by adding the new definition of **PS1**. If you have root permission, you can modify the **PS1=** line in the `/etc/profile` file. On some distributions (Red Hat 5.1 at least), the `/etc/bashrc` file overrides the `/etc/profile` setting of **PS1** and **PS2**.

My default prompt includes my user name "giles", the name of my work machine "nikola" and my home directory `/home/giles`. The simplest prompt is a single character. I can change my default prompt to a simple `$` by typing:

```
[giles@nikola giles]$ PS1="$ "
```

I use the quotes to force a space after the prompt, making it more readable.

Escape Sequences

Bash 2.02 Man Page

Many escape sequences are offered by the Bash shell for insertion in the prompt. See the sidebar which shows the Bash 2.02 man page.

```
$ PS1="\u@\h \w> "<\n>
giles@nikola giles>
```

This example creates a prompt that is close to the default on most Linux distributions. I wanted a slightly different appearance, so I changed it to include the time by typing:

```
giles@nikola giles> PS1="[ ][\u@\h:\w]$ "<\n>
[21:52:01][giles@nikola:~]$
```

Bash also provides an environment variable called **PROMPT_COMMAND**. The contents of this variable are executed as a regular Bash command just before Bash displays a prompt.

```
[21:55:01][giles@nikola:~] PS1="[ ][\u@\h:\w]$ "<\n>
[giles@nikola:~] PROMPT_COMMAND="date +%H%M"
2155
[giles@nikola:~] ls
bin mail
2156
[giles@nikola:~]$ unset PROMPT_COMMAND
[giles@nikola:~]
```

In this example, I changed **PS1** by eliminating the escape sequence, so that time was no longer a part of the prompt. Then I used **date +%H%M** to display the time in a format I like better. At the end, I used the **unset** command to remove the **PROMPT_COMMAND** environment variable.

External Commands

As I discuss the use of external commands in prompts, I'll use the **\$(command)** convention for command substitution; that is,

```
$(date +%H%M)
```

means “substitute the output from the **date +%H%M** command here.” This works in Bash 1.14.7 and 2.0+. In some older versions of Bash, you may need to use backquotes (``date +%H%M``). Backquotes can be used in Bash 2.0+ but are being phased out in favor of **\$()**, which nests better. If you are using an earlier version of Bash, substitute backquotes wherever you see **\$()**. If the command substitution is escaped (i.e., `\$(command)`), then use backslashes to escape BOTH your backquotes (i.e., `\`command\``).

You don't want to insert much material from an external command into the prompt, as a prompt of great length may be created. You also want to use a fast command, because it will be executed each time your prompt appears on the screen. Delays in the appearance of the prompt while you are working can be annoying.

```
[giles@nikola:~]$ PS1="\$(date +%H%M)][\u@\h:\w]$ "[2159][giles@nikola:~]$
```

Note the backslash before the dollar sign of the command substitution. Without it, the external command is executed exactly once: when the **PS1** string is read into the environment. For this prompt, it would display the same time no matter how long the prompt was used. The backslash prevents immediate shell interpretation of the command, so **date** is called each time a prompt is generated.

Shell Scripts

Linux comes with many small utility programs such as **date**, **grep** and **wc** which allow you to manipulate data. If you wish to create complex combinations of these programs within a prompt, it may be easier to make a shell script and call it from the prompt. An example of a small shell script used within a prompt is given in Listing 1.

Listing 1. Shell Script for Use in Prompt

I keep this as a shell script in my `~/bin` directory, which is in my path. Use it in a prompt in this way:

```
[2203][giles@nikola:~]$ PS1="\u@\h:\w (\$(lsbytesum) Mb)\$ "[giles@nikola:~ (0 Mb)]$ cd /bin  
[giles@nikola:/bin (4.498 Mb)]$
```

Non-Printing Escape Sequence

Non-printing escape sequences can be used to produce interesting effects in prompts. To use these escape sequences, you need to enclose them in `\[` and `\]`, telling bash to ignore this material while calculating the size of the prompt. Failing to include these delimiters results in line editing code placing the cursor in the wrong place, because it doesn't know the actual size of the prompt. Escape sequences must also be preceded by `\033[` in bash prior to version 2 or by either `\033[` or `\e[` in later versions.

xterm title bar

This example modifies the title bar of an xterm window. If you try to change the title bar of an xterm with your prompt when you are at the console, you'll produce garbage in your prompt. To avoid this problem, test the `TERM` environment variable to determine if your prompt is going to be in an xterm. If the shell is an xterm, the shell variable (`${titlebar}`) is defined. It consists of the appropriate escape sequences, and `\u@\h:\w`, which puts *user@machine: working directory* in the xterm title bar. This is particularly useful with minimized xterms, making them more rapidly identifiable. The other material in this prompt should be familiar from previous prompts we've created.

Listing 2. Function to Set Titlebar

Listing 2 is a function that can be incorporated into `~/.bashrc`. The function name can then be called to execute the function. The function, like the `PS1` string, is stored in the environment. Once the `PS1` string is set by the function, you can remove the function from the environment by typing `unset prom1`. Since the prompt can't change from being in an xterm to being at the console, the `TERM` variable isn't tested each time the prompt is generated.

I used continuation markers (backslashes) in the definition of the prompt to allow it to be continued on multiple lines. This improves readability, making it easier to modify and debug.

I define this as a function because this is how the Bash Prompt package deals with prompts: it is not the only way to do it, but it works well. As the prompts you use become more complex, it becomes more and more cumbersome to type them in at the prompt and more practical to create them in a text file. To test this example at the prompt, save the function as a text file called "prom1". The Bash `source` command can be used to read the prompt function by typing:

```
[giles@nikola:~ (0 Mb)]$ source prom1
```

To execute the prompt function, type:

```
[giles@nikola:~ (0 Mb)]$ proml
```

Color Text

As mentioned before, non-printing escape sequences must be enclosed in `\[033[and \]`. For color escape sequences, they must also be followed by a lowercase `m`. To include blue text in the prompt:

```
PS1="\[\033[34m\][\$(date +%H%M)][\u@\h:\w]$
```

The blue color that starts with the 34 color code is never switched back to the regular gray, so any text you type after the prompt is still in the color of the prompt. This is also a dark shade of blue (very hard to read), so combining it with the bold code might help:

```
PS1="\[\033[1;34m\][\$(date +%H%M)][\u@\h:\w]$ \[\033[0;37m\] "
```

The prompt is now in light blue, and it ends by switching the color back to gray, which is the color most of us expect when we type.

Bash Color Equivalences

Background colors can be set by using 44 for Blue background, 41 for a Red background, etc. No bold background colors are available. Combinations can be used, e.g., Light Red text on a Blue background: `\[033[44;1;31m\]`. Other codes available include 4 for Underscore, 5 for Blink, 7 for Inverse and 8 for Concealed.

Listing 3. elite Function

The prompt I use most of the time is based on one called “elite2” in the Bash Prompt package, which I have modified to work better on a standard console (Listing 2). (The original uses special xterm fonts.) I define the colors as temporary shell variables for the sake of readability—it is easier to work with. The **GRAD1** variable is a check to determine what terminal you are on, and it needs to be done only once. The prompt you see looks like Figure 1.

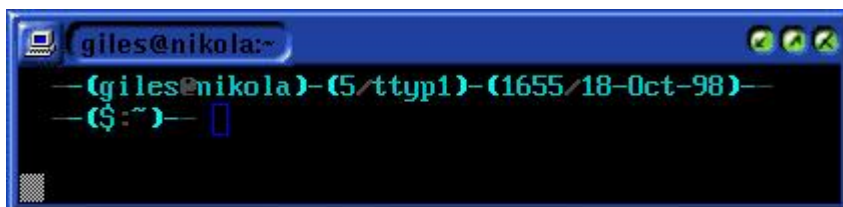


Figure 1. My Prompt

The Bash Prompt package is available in beta at <http://bash.current.nu/> and is the work of several people, co-ordinated by Rob Current. The package offers a

simple way to use multiple prompts or “themes”. Several of these prompts use the extended VGA character set, so they look bad unless used with special xterm fonts. The “fire” theme shown in Figure 2 requires these fonts. See *Stumpy's ANSI Fonts* page at <http://home.earthlink.net/~us5zahns/enl/ansifont.html> for instructions on installing and using these fonts.



Figure 2. Fire Prompt from the Bash Prompt Package

You can change the prompt in your current terminal using the example elite function by typing **source elite** (assuming the elite function file is in your path) followed by **elite**. This leaves you with an extra function (elite) in your environment space—if you want to clean up the environment, type **unset elite**.

This would seem like an ideal candidate for a small shell script, but a script doesn't work here because a script cannot change the environment of your current shell—it can change only the environment of the subshell it runs in. Environment variables of your current shell can be changed by environment functions. The Bash Prompt package puts a function called **callbashprompt** into your environment, and while they don't document it, it can be called to load any Bash Prompt theme on the fly. It looks in the theme directory it installed, sources the function you requested, loads it, then unsets the function. **callbashprompt** wasn't intended to be used this way and has no error checking, but it works quite well.

Resources

Giles Orr is a Systems Librarian at Georgia College and State University. He's been using Linux for four years. He doesn't claim to be a master programmer and would welcome improvements to the code in this article. He can be reached at giles@interlog.com. He is the maintainer of the Bash Prompt HOWTO at <http://metalab.unc.edu/LDP/HOWTO/Bash-Prompt-HOWTO.html>.

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

[Advanced search](#)

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

[Advanced search](#)

A High-Availability Cluster for Linux

Phil Lewis

Issue #64, August 1999

Mr. Lewis tells us how he designed and implemented a simple high-availability solution for his company.

Although Linux is known to be an extremely stable operating system, the fact that the standard PC hardware is not quite so reliable must not be overlooked. I have been maintaining Linux servers for a long time, and in most cases when a system has failed, it has been due to server hardware failure. UNIX in the commercial world is known for having good clustering and high-availability (HA) technologies.

In my present company, Electec, we rely heavily upon e-mail (Sendmail and IMAP4), Windows file sharing (Samba), FTP and dial-up authentication (**radius**) services on a 24-hour basis for communication with our suppliers, staff and customers who are located in different time zones. Until recently, all of these services were consolidated on one Linux server. This system had served us very well. However, it was just a matter of time before a hardware failure occurred, which would cause us loss of productivity and revenue.

High availability is becoming increasingly important as we depend more and more on computers in business. I decided to design and implement an inexpensive high-availability solution for our business-critical needs without requiring the use of expensive additional hardware or software. This article covers the design aspects, pitfalls and implementation experiences of the solution.

Clusters or Fault-Tolerant Hardware

Quite a few different approaches and combinations of approaches exist for high availability on servers. One way is to use a single, fault-tolerant server with redundant power supplies, RAID, environmental monitoring, fans, network interface cards and so on. The other way involves the use of several units of

non-redundant hardware arranged in a cluster, so that each node (or server) in the cluster is able to take over from any failures of partner nodes. The fault-tolerant server approach has the advantage that operating system, application configurations and operations are the same as if you were using a simple, inexpensive server. With a cluster, the application and OS configurations can become very complex and much advanced planning is needed.

With a fault-tolerant server, failures are taken care of in such a way that clients do not notice any downtime—recovery is seamless. Ideally, this should also be the case with node failures in a cluster. In many cases, the hardware cost of a cluster is far less than that of a single fault-tolerant server, especially when you do not have to spend a great deal of money for one of the commercial cluster software offerings.

Trade-off Between Cost and Downtime

A trade-off is present between cost and client disruption/downtime. You must ask yourself how much downtime you and your users can tolerate. Shorter downtimes usually require a much more complex or costly solution. In our case, I decided we could live with approximately five minutes downtime in the event of a failure; therefore, I chose to use a cluster as opposed to a single fault-tolerant server.

Many clustering solutions are available to the UNIX market, which can provide almost zero downtime in the event of a node takeover by means of session and network connection takeover. These solutions are mostly expensive and normally require the use of external shared storage hardware. In our case, we can allow for sessions and connections to be lost. This simplifies the task of implementing a high-availability cluster.

Distribution of Storage Devices

When implementing HA clustering, some recommend that a shared storage device, such as a dual-port RAID box, be used. However, it is possible to approach the problem by using a separate storage device for each node in the cluster and mirroring the storage devices when necessary. Each avenue has its own merits. The shared storage approach has the benefit of never requiring any software mirroring of data between cluster nodes, thus saving precious CPU, I/O and network resources. Sharing is also beneficial because data accessible from another cluster node is always up to date. The mirroring approach, which uses separate storage devices, has the advantage that the cluster nodes do not have to be in the same geographical location and are therefore more useful in a disaster recovery scenario. In fact, if the mirror data was compressed, it could be sent over a WAN connection to a remote node.

RAID systems are available, which allow the disks to be geographically distributed and which require interconnectivity by optical fiber; however, these are rather expensive. Two sets of simple storage devices are less expensive than a dual-port RAID box of similar capacity. The dual-port RAID box can, in some cases, introduce a single point of failure in the cluster. If the RAID file system is somehow corrupted beyond recovery, it would cause serious cluster downtime. Most RAID systems mirror the data at the device level and have no regard for which file system is in use. A software-based system can mirror files in user space, so if a file becomes unreadable on one node it will not necessarily copy the same file system corruption to the other node. Due to this advantage and the cost factor, I decided to use separate storage devices on each node in the cluster. It should be noted that even if a dual-ported storage device is used, both nodes in the cluster should never mount the same partition read/write simultaneously.

Cluster Load Balancing

Spreading the workload evenly across the nodes in a cluster is preferable to having one node do all the work until it fails, then having another node take over. The term for this is a *hot standby* system.

Load balancing can take many forms, depending on the service in question. For web servers, which provide simple static pages and are often read-only, a round-robin DNS solution can be quite effective. Unfortunately, with the read/write or transactional type of services such as e-mail or database access, unless the connection and session information from the service on one node can be shared and used by other nodes, it would be very difficult to provide seamless load balancing over the cluster. It would also require the disk mirroring to be near-instantaneous and use lots of distributed locking techniques, which most daemons will not support without complex modifications. To avoid these drawbacks, I decided to use a simpler approach which stands between the hot standby and the network-level load balancing.

In my two-node cluster, I put half of the required services on node "A" (serv1) and the other half on node "B" (serv2). A mutual failover configuration was employed so that if node A failed, node B would take over all of its services and vice versa.

Service Suitability for Clustering

We had to decide which services needed to be running on the overall cluster. This involved comparatively rating how much computing resource each service would consume. For our previous Linux server, it was found that Samba and Cyrus IMAP4 were the most resource-intensive services with FTP, httpd and Sendmail following close behind. Careful consideration had to be given to

which services were suited to running on two or more nodes concurrently. Examples of such services included Sendmail (for sending mail only or as a relay), bind, httpd, ftpd (downloading only) and radius. Examples of services which cannot be run in such a way are Cyrus IMAP4, ftpd (uploading) and Samba. Samba cannot be run on two servers at once, as that would result in two servers broadcasting the same netbios name on the same LAN. It is not yet possible to have PDC/BDC (primary and backup domain controller) arrangements with the current stable versions of Samba. The pages on a simple web site, on the other hand, do not change often. Therefore, mirroring is quite effective and parallel web servers can run quite happily without major problems. The servers were configured so that each one took primary care of a specific group of services. I put Samba on serv1 and Cyrus IMAP4 on serv2. The other services are shared in a similar way; httpd, bind and radius run on both nodes concurrently.

Node Takeover

In the event of a node failure, the other node takes over all the services of the failed one in such a way as to minimize disruption to the network users. This was best achieved by using IP (Internet protocol) and MAC (mandatory access control) address takeover from the failed node onto an unused Ethernet card on the takeover node. In effect, the node would appear to be both serv1 *and* serv2 to the network users.

The use of MAC address takeover was preferred in order to avoid potential problems with the clients' ARP (address resolution protocol) cache still associating the old MAC address with the IP address. MAC address takeover, in my opinion, is neater and more seamless than IP takeover alone, but unfortunately has some scalability limitations.

Networking Hardware

While considering the HA network setup, it was very important to eliminate all possible single points of failure. Our previous server had many single points of failure; the machine itself, the network cable, the Ethernet hub, the UPS, etc. The list was endless. The network has been designed to be inexpensive and reliable as shown in Figure 1.

Figure 1. Network Diagram for Our Two Nodes

The network diagram shows the three network interface cards (NICs) in each server. The first NIC in each server is used for the main LAN access to clients. Each node is plugged in to a separate Ethernet switch or hub to give redundancy in case of a switch lockup or failure. (This actually happened to us not so long ago.) The second NIC is used for creating a private inter-node

network using a simple cross-over 100BaseTX full-duplex Ethernet cable. A cross-over cable is far less likely to fail than two cables plugged into an Ethernet hub or switch. This link is primarily used for disk mirroring traffic and the cluster heartbeat. It also helps take the network traffic load off the main interface and provides a redundant network path between the nodes. The third NIC is the redundant LAN access card used for MAC and IP address takeover in the event of a remote node failure in the cluster. Again, they are plugged in to different Ethernet switches or hubs for greater network availability.

Cluster Partitioning

If a node fails in some way, it is vital that only one of the nodes performs the IP and MAC address takeover. Determining which node has failed in a cluster is easier said than done. If the heartbeat network failed while using a simplistic takeover algorithm, both of the nodes would wrongly perform MAC, IP and application takeover and the cluster would become partitioned. This would cause major problems on any LAN and would probably result in some kind of network and server deadlock. One way to prevent this scenario from taking place is to make the node which first detects a remote node failure, remote login to each of that remote node's interfaces and put it into a standby run level (e.g., single-user mode). This run level would prevent the failed node from attempting to restart itself and thus stop an endless failure-recovery loop. There are problems with this method. What if node A (which has a failed NIC) thinks node B is not responding, then remotely puts node B into single-user mode? You would end up with no servers available to the LAN. There must be a mechanism to decide which node has actually failed. One of the few ways to do this on a two-node cluster is to rely on a third party. My method of implementing this is to use a list of locally accessible devices which can be pinged on the LAN. By a process of arbitration, the node which detects the highest number of unreachable devices will gracefully surrender and go into the standby runlevel. This is shown in Figure 2.

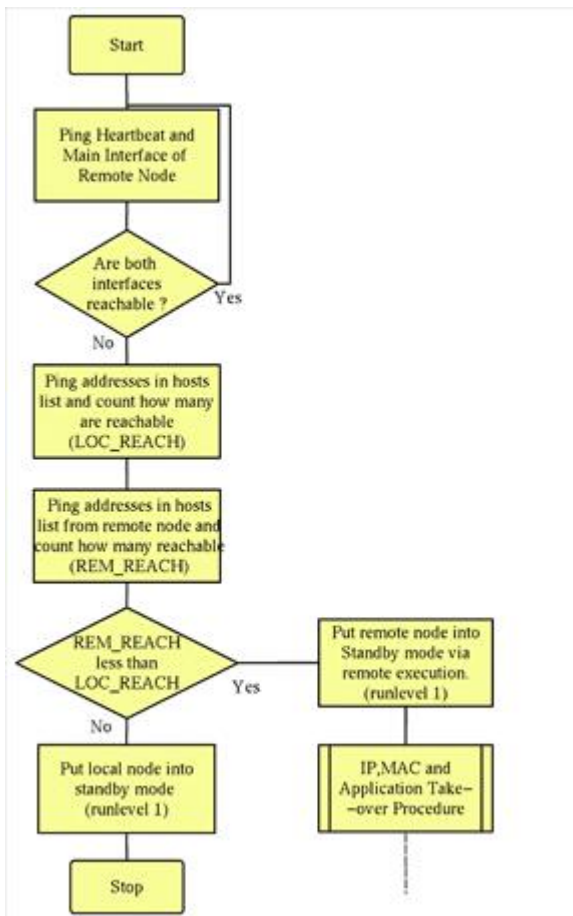


Figure 2. Flow Chart Showing Overview of Cluster Daemon Algorithm

Distributed Mirroring of File Systems

To implement this solution with minimal risk of data loss, the data on the two servers must be constantly mirrored. It would be ideal if the data written to serv1 was simultaneously written to serv2 and vice versa. In practice, a near-perfect mirror would require a substantial kernel implementation with many hurdles along the way, such as file system performance and distributed lock management. One method would be to implement a RAID mirror which used disks from different nodes: a cluster file system. This is supposed to be possible in later incarnations of the 2.1 and probably the 2.2 kernel by using md, NFS and network block devices. Another solution, which also remains to be evaluated, is the use of the CODA distributed file system.

Synchronization Design

A practical way to have a mirror of data on each node is to allow the frequency of the file mirroring to be predefined by the administrator, not only for nodes but rather on a per file or directory basis. With this fine-grained level of control, the data volatility characteristics of a particular file, directory or application can be reflected in frequency of mirroring to the other node in the cluster. For example, fast-changing data such as an IMAP4 e-mail spool, where users are constantly moving, reading and deleting e-mail, could be mirrored every

minute, whereas slow-changing data such as the company's mostly static web pages could be mirrored hourly.

Trade-off Between Mirror Integrity and Excessive Resource Usage

Trade-offs must be considered when mirroring data in this way. One major trade-off is mirror integrity with CPU and I/O resource consumption. It would be nice if I could have my IMAP4 mail spools mirrored each second. In practice, this would not work because the server takes 15 seconds to synchronize this spool each time. The CPU and disk I/O usage could be so high that the services would be noticeably slowed down. This would seem to defeat the objective of high availability. Even if the CPU had the resources to read the disks in less than one second, there might still be problems transferring the data changes between the nodes due to a network throughput bottleneck.

The Risk of Data loss

This mirroring approach does have flaws. If a file is saved to a Samba file share on serv1, and serv1 fails before they are mirrored, the file will remain unavailable until serv1 fully recovers. In a worst-case scenario, the serv1 file system will have been corrupted and the file lost forever. However, compared to a single server with a backup tape, this scenario is less risky because traditional backups are made far less frequently than the mirroring in the cluster. Of course, a cluster is no replacement for traditional backups which are still vital for many other reasons.

Resynchronization of Files on Node Recovery

A major design factor is resynchronization (mirroring back) of the files once a failed node has recovered. A reliable procedure must be employed so that data which has changed on the failover node during the failure period is mirrored back to the original node and not lost, because the original node overwrites or deletes it in the restoration procedure. The resynchronization procedure should be implemented so that a node cannot perform any mirroring while another node has taken over its services. Also, before the services can be restarted on the original node, all files associated with it must be completely mirrored back to this original node. This must be done while the services are off-line on both nodes to prevent the services from writing to the files being restored. Failure to prevent this could result in data corruption and loss.

Mirroring Warnings

The main problem when using this solution was with IMAP4 and pop3 mail spools. If an e-mail message is received and delivered on serv2, and serv2 fails before mirroring can take place, serv1 will take over the mail services.

Subsequent mail messages would arrive in serv1's mail spool. When serv2 recovers, any e-mail received just before failure will be overwritten by the new mail received on serv1. The best way to solve this is to configure Sendmail to queue a copy of its mail for delivery to the takeover node. In the event that the takeover node is off-line, mail would remain in the Sendmail queue. Once the failed node recovered, e-mail messages would be successfully delivered. This method requires no mirroring of the mail spools and queues. However, it would be necessary to have two Sendmail configurations available on both nodes: one configuration for normal operation and the other for node takeover operation. This will prevent mail from bouncing between the two servers.

I am not a Sendmail expert. If you know how to configure dual-queuing Sendmail delivery, please let me know. This part is still a work in progress. As a temporary measure, I create backup files on resynchronization of the mail spool with manual checking on node recovery, which is quite time consuming. I also prevent such difficulties by mirroring the mail spool as frequently as possible. This has an unfortunate temporary side effect of making my hard disks work overtime. Similar problems would be encountered when clustering a database service. However, a few large UNIX database vendors are now providing *parallel* versions of their products, which enable concurrent operation across several nodes in a cluster.

The Node Recovery Procedure

A node could fail for various reasons ranging from an operating system crash, which would result in a hang or reboot, to a hardware failure, which could result in the node going into standby mode. If the system is in standby mode, it will not automatically recover. The administrator must manually remove a standby lock file and start run-level 5 on the failed node to confirm to the rest of the cluster that the problem has been resolved. If the OS hangs, this would have the same effect as a standby run level; however, if the reset button is pressed or the system reboots, the node will try to rejoin the cluster, as no standby lock file will exist. When a node attempts to rejoin the cluster, the other node will detect the recovery and stop all cluster services while the resynchronization of the disks takes place. Once this has completed, the cluster services will be restarted and the cluster will once again be in full operation.

Implementation Platform

My choice of Linux distribution is Red Hat 5.1 on the Intel platform. There are, however, no reasons why this could not be adapted for another Linux distribution. The implementation is purely in user space. No special drivers are

required. Some basic prerequisites are necessary in order to effectively deploy this system:

- Two similarly equipped servers, especially in terms of data storage space, are needed.
- Three network interface cards per server are recommended, although two might work at the expense of some modifications and extra LAN traffic.
- Sufficient network bandwidth is needed between the cluster nodes.

My system consists of two Dell PowerEdge 2300 Servers, each complete with:

- three 3C905B 100BaseTX Ethernet cards
- two 9GB Ultra SCSI 2 hard disks
- one Pentium II 350 MHz CPU

Figure 3. Photograph of the Two-node Cluster

Overview of Cluster Software Configuration

The administrator can configure groups of files which are mirrored together by creating small mirror description files in the configuration directory. Note directory entries must end with a forward slash (/). Below is the description file for my lpd mirroring, /etc/cluster.d/serv1/Flpd:

```
/var/spool/lpd/  
/etc/printcap
```

The frequency of the mirroring is then controlled by an entry in the /etc/crontab file. Each entry executes the **sync-app** program, which examines the specified service mirror description file and mirrors the contents to the specified server IP address. In this example, the specified server address is the cross-over cable IP address on serv2. Mirroring of the lpd system is done every hour. These crontab entries are from serv1:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * root \  
/usr/local/bin/sync-app /etc/cluster.d/serv1/Fsmbd \  
serv2-hb  
0 * * * * root /usr/local/bin/sync-app \  
/etc/cluster.d/serv1/Flpd serv2-hb
```

Cluster Daemon Implementation

The brains of the system lie in the cluster daemon, **clusterd**. This was written in the Bourne shell and will soon be rewritten in C. The algorithm outline is shown as a flow chart in Figure 2.

clusterd continuously monitors the ICMP (Internet control message protocol) reachability of the other node in the cluster, as well as a list of hosts which are normally reachable from each node. It does this using a simple **ping** mechanism with a timeout. If the other node becomes even partially unreachable, **clusterd** will decide which node actually has the failure by counting the number of hosts in the list which each node can reach. The node which can reach the fewest hosts is the one which gets put into standby mode. **clusterd** will then start the failover and takeover procedures on the working node. This node then continues to monitor whether the failed node recovers. When it does recover, **clusterd** controls the resynchronization procedure. **clusterd** is invoked on each node as:

```
clusterd <local-nodename> <remote-nodename>
```

It has to know which applications and services are running on each node so that it knows which ones to start and stop at failover and takeover time. This is defined in the same configuration directories as the service mirror description files discussed earlier. The configuration directories in each node are identical and mirrored across the whole cluster. This makes life easier for the cluster administrator as he can configure the cluster from a single designated node. Within the `/etc/cluster.d/` directory, a `nodename.conf` file and a `nodename` directory exist for each node in the cluster. The `reachlist` file contains a list of reachable external hosts on the LAN. The contents of my `/etc/cluster.d` directory are shown here:

```
[root@serv1 /root]# ls -al /etc/cluster.d/
total 8
drwxr-xr-x  4 root root 1024 Nov 15 22:39 .
drwxr-xr-x 23 root root 3072 Nov 22 14:27 ..
drwxr-xr-x  2 root root 1024 Nov  4 20:30 serv1
-rw-r--r--  1 root root  213 Nov  5 18:49 serv1.conf
drwxr-xr-x  2 root root 1024 Nov  8 20:29 serv2
-rw-r--r--  1 root root  222 Nov 22 22:39 serv2.conf
-rw-r--r--  1 root root   40 Nov 12 22:19 reachlist
```

As you can see, the two nodes are called `serv1` and `serv2`. The configuration directory for `serv1` has the following files: `Fauth`, `Fclusterd`, `Fdhcpd`, `Fipd`, `Fnamed`, `Fradiusd`, `Fsmbd`, `K10radiusd`, `K30httpd`, `K40smb`, `K60lpd`, `K70dhcpd`, `K80named`, `S20named`, `S30dhcpd`, `S40lpd`, `S50smb`, `S60httpd` and `S90radiusd`.

Files beginning with the letter `F` are service mirror description files. Those starting with `S` and `K` are linked to the SysVinit start/stop scripts and behave in a similar way to the files in the SysVinit run levels. The `S` services are started when node `serv1` is in normal operation. The `K` services are killed when node `serv1` goes out of service. The number following the `S` and `K` determine the order of starting and stopping the services. **clusterd**, running on node `serv2`, uses this same `/etc/cluster.d/serv1/` directory to decide which services to start on `serv2` when node `serv1` has failed. It also uses the `serv1` service mirror

description files (those files starting with F) to determine which files and directories need to be mirrored back (resynchronized) to serv1 after it has recovered.

The configuration directory for node serv2 contains Fftpd, Fhttpd, Fimapd, Fsendmail, K60sendmail, K80httpd, K85named, K90inetd, S10inetd, S15named, S20httpd and S40sendmail. As you can see, the serv2 node normally runs Sendmail, named, httpd, IMAP4 and ftpd.

Network Control Scripts

Whenever the network interfaces need to be brought up or down I have used Red Hat's supplied **ifup** and **ifdown** scripts. This makes the network interface configuration more tightly integrated with the GUI network configuration tools. The node configuration files, */etc/cluster.d/nodename.conf*, allow you to specify the Ethernet NIC and its purpose on each node in the cluster. My two node configuration files are shown in Listings 1 and 2.

Listing 1. serv1 Configuration File

Listing 2. serv2 Configuration File

To implement the MAC address take over, one important addition must be made to the Red Hat Ethernet configuration files. You must add a line to the */etc/sysconfig/network-scripts/ifcfg-eth2* file to set the MAC address. eth2 is the redundant interface in my case, so I need it to take over the MAC address of the main service interface on the other node in the cluster. In other words, the MAC address of eth2 on serv2 must be the same as the MAC address of eth0 on serv1. The line 'MACADDR=00:10:4B:63:1C:08' was appended to this file on node serv2. The Red Hat ifup command will use this variable when bringing up an interface. A similar modification must be made to each node.

If you use an Ethernet switch (instead of a hub), it will be necessary to set the MAC address cache timeout to a suitable period to avoid the cluster losing communication with the LAN clients after a MAC address takeover. I set ours to 20 seconds for the ports which are connected directly to the nodes. Consult your switch manual or vendor if you need information on how to do this. It can usually be done via the console cable.

Centralized Cluster Administration

I have created service mirror description files and crontab entries for */etc/hosts*, *passwd/group* files and the entire */etc/clusterd/* directory so that I can administer the cluster from a single node. This greatly simplifies cluster configuration. To avoid confusion, I found it helpful to create a DNS alias for

each service used on the cluster which points to the primary node for that service. Thus, when I need to configure Samba, all I need to do is remotely log in to `samba.yourdomainname.com`. If the secondary node for a service is configured by mistake, any changes will be ignored until the primary node fails.

Current Software Limitations

Currently for my system, only two nodes may be in a cluster. Scaling this up to clusters of more than two nodes should not be difficult, although instead of MAC address takeover, a different approach will probably have to be used because of the large number of NICs required for larger clusters.

Other Utilities Used

Several useful utilities enabled me to do efficient mirroring. **rsync** is an invaluable utility which uses the rsync algorithm. This program will look for changes in files and mirror only the parts which have changed rather than the whole file. It also checks if the file has been updated by examining the modification date and file size before doing any further comparisons. **ssh** (secure shell) can also be used between the nodes in conjunction with *rsync* so that the mirrored data is sent via an encrypted and authenticated connection. Alternatively, you can use **rsh** if you prefer.

When rsync is doing file comparisons, it uses the file's date and time; therefore, it is vital that the nodes all agree on the same time. I chose to run the **netdate** utility every hour from cron. The nodes used a list of remote trusted time sources. To make sure a failed node boots with the correct time, the CMOS PC clock is updated after running netdate.

Synchronization Implementation

rsync was configured so that files in the source directory which do not exist on the target directory are deleted. This behaviour is necessary to avoid accumulative and excessive disk usage on the target node. If this setting is not used, a user connected to a Samba file share would effectively not be able to delete any file on the mirrored node. The same goes for almost all applications. **clusterd** is configured to create backups of deleted or changed files when the resynchronization procedure is in progress. This can help minimize the risk of data loss in the event of mirroring failure prior to a node takeover. Subsequent removal of backup files would necessitate some human intervention, after it has been confirmed that files or data were not lost after the node recovery. This is done using the **--backup** option of rsync version 2.2.1. You may find it more CPU efficient to turn off the rsync algorithm and fully mirror files which have changed instead of mirroring the changes; however, this will utilize more network bandwidth.

Resynchronization Implementation

The resynchronization (mirroring back) procedure was implemented using `rsync`, which uses a lock file to disallow any mirroring to another node when a node failure is sensed. The lock file is checked for existence by `sync-app` before any files are mirrored. This prevents node A mirroring to node B, while node B is mirroring the same files to node A.

Using a Shared Storage Device

If preferred, `clusterd` could be used with a shared and/or distributed storage device by removing the resynchronization function and by not using `sync-app`, although I have not tried this.

Testing and Results

To test server failure, I had to simulate the failure of every interface on the cluster. In each case, the cluster took the expected action and shut down the correct server. In the case of the inter-node/heartbeat network failing, the nodes simply carried on normal operation and notified the administrator of the failure. On a point-to-point network of this nature, it is almost impossible to determine which NIC is at fault. I simulated various network switch failures and power supply failures. The results were all as expected. After a node was put into standby (single-user) mode, I had to manually remove a standby lock file in order to fully bring up the node again. If a node recovered and entered a network runlevel while the standby lock file still existed, the remote node immediately put the node back into standby mode to prevent an IP and MAC address clash on the LAN.

Mirroring was tested over a period of several months, and I found that the nodes could typically compare 6GB of unchanged data in approximately 50,000 files in under 45 seconds.

After catastrophic node failure (I pulled the power plug from the UPS), recovery time for the node was around 10 to 15 minutes for `fsck` disk checking, and a disk resynchronization time of around three minutes (9GB of data). This represented a cluster services downtime of around three minutes to the LAN clients.

Failover delay from when a node failed until the remote node fully took over was typically 60 to 80 seconds. The effect on users depended on the service: Sendmail, IMAP4, http and FTP simply refused connection for users for the duration, whereas Samba sometimes momentarily locked up a Windows PC application when files were open at the point of failure. `radius` and `dhcpcd` caused no client lock-outs, probably because of their UDP implementation.

Conclusions

On the whole, the cluster provides us with much better system availability. It is a vast improvement over the single server, as we can now afford to do server maintenance and upgrades during working hours. We have not yet had any catastrophic failures with the new Dell servers, but the test results show a minimal downtime of less than two minutes while a node takes over. We have saved large amounts of capital by implementing a simple high-availability cluster without the need for expensive specialist hardware such as dual ported RAID.

This clustering solution is certainly not as advanced as some of the commercial clusters or as thorough as some of the upcoming open source Linux-HA project proposals; however, it does sufficiently meet our needs.

The system has been in full-time production operation since September 1998. We have over 30 LAN clients using the cluster as their primary "server". The system has proven to be reliable. The company sees the server as a business-critical system, and we have achieved the objectives of high availability.



Philip Lewis is from the UK and graduated from the University of Birmingham in 1994. He has spent three years working in Singapore and now runs his own consultancy company in UK designing WAN/LAN infrastructures and writing Linux software. His interests include Linux software development and hacking, telecommunications, network security, promoting Linux, making wine and eating good food in Malaysia. He can be reached via e-mail at lewisplj@email.com.

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

[Advanced search](#)

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

Introduction To Sybase, Part 3

Jay Sissom

Issue #64, August 1999

Writing a Sybase web application.

Welcome Back! In the first two parts of this series, we installed the Sybase database and configured the Perl language to work with the database. In part three, we will create a web application using our newly installed database server. Before you can manage an application using the Sybase server, some background information is in order. I will assume you have read the two previous articles in the series.

The example program is a bookstore on the Web. We will discuss all aspects of creating and maintaining a client-server application using the Sybase SQL server. You won't be an expert after reading the article, but you'll have a good idea of what needs to be done for client-server applications. The example application will not be able to compete with Amazon.com, but it will give you an idea of how to design a client-server application. We'll first take a look at some Sybase SQL server basics.

Transactions

The Sybase SQL Server is a transactional database server. A transaction is a single piece of work. For example, placing an order would be considered one transaction. Moving money from one account to another is one transaction. Even if a transaction affects two tables in our database, the entire transaction should be completed or should not happen at all—it should never be half-completed. For example, if you want to transfer \$100 from your savings account to your checking account, you either want the transfer to happen or not happen. You don't want the \$100 to be removed from your savings account but not placed in your checking account. The SQL to accomplish this transfer would look something like this:

```
update accounts_t set balance = balance - 100
  where account_nbr = 'mysaving'
```

```
update accounts_t set balance = balance + 100
  where account_nbr = 'mychecking'
```

The bank wouldn't want the first update to execute without the second one. If it were my account, I know I wouldn't. The Sybase SQL Server allows the developer to denote where a transaction begins and where it ends.

```
begin transaction
  update accounts_t set balance = balance - 100
    where account_nbr = 'mysaving'
  if @@error != 0
  begin
    rollback transaction
    return
  end
  update accounts_t set balance = balance + 100
    where account_nbr = 'mychecking'
  if @@error != 0
  begin
    rollback transaction
    return
  end
  commit transaction
  if @@error != 0
  begin
    rollback transaction
  end
```

If you do not use the `begin transaction` and `commit transaction` commands, the database server will assume each SQL command is a single transaction. Make sure you use transaction control in your application where it is needed. For more information on transactions, see Chapter 17 in the *Transact-SQL User's Guide*.

Logs

The SQL Server uses a log to keep track of all transactions. Each database has its own log. The log is part of the database and is not human readable. Each change made to a database is saved in the database's transaction log. Some databases save their data and log information in the same area. Most databases have separate areas for the data and their log. Each area of a database is called a segment. By default, a database has three segments:

- System: stores the database's system tables.
- Log: stores the database's transaction log.
- Default: stores all other database objects.

You can create a database where all three segments are on the same device. If you do this, you won't have as much control over the database as you'd like. The best way to create a database is to separate the log segment from the system and default segments. When you create the database, you specify which devices to put the database on. For more information on these segments and on creating your own custom segments, look in chapter 16 of the *System Administration Guide* (see Resources). For more information on creating

databases, look in chapter 14. In our example program, a database called `book_d` is created with 20MB allocated for data and 10MB for log.

All transactions that modify a database are saved in the database's transaction log. The data in this log will keep increasing until you back up the log to tape or disk. The data in the log will remain even after all transactions are committed. It stays until the log segment is backed up. This means you need to back up your database often, or this log segment will fill up. When the log segment fills up, you cannot make changes to data in this database until you back up the log segment.

A function is available that will allow the database server to automatically call a stored procedure when a segment is filling up. So, you can write a stored procedure to automatically back up the log when the segment is close to full. A stored procedure is an object in the database containing SQL code. An example stored procedure called **`sp_thresholdaction`** that will back up the log to disk is included in the application. For more information about Threshold procedures, see chapter 21 of the *System Administration Guide*.

Backups

Just as any other part of a computer system, databases need to be backed up. One approach would be to shut down the database server, then back up the database device files. This could work, but Sybase says a back up made this way is not guaranteed to restore properly. Each database has to keep track of its logs and the sequence of transactions. If any of these are out of sync with the others, Sybase cannot use the database. Fortunately, Sybase has created a backup server to help you back up your databases. It will even let you run your backups while the database is still up. Your bookstore can stay up 24 hours a day, 7 days a week.

You can back up your databases and logs to either disk or tape using the Sybase backup server. To back up to tape, you'll need to configure your tape device. Read Chapter 18 of the *System Administration Guide* to learn how to configure your tape drive.

Depending on the size of your databases, I would recommend backing up the logs at least once during the day and the databases every night. If your databases are huge, you might not be able to do this. Back up the logs as often as you can, but think of log backups as changes made to the database. If the database gets corrupted, you'll need to restore the database, then all the logs you backed up since the database backup was made.

If you choose to save logs to a disk drive, I would highly recommend they be saved on a separate disk from the one holding your database. On one of our

production database servers, we back up the logs to disk at noon each day, then FTP them to another system. This system has 20 1GB drives on it, so databases and logs are located on different drives, and the log backups are on yet another disk. For best performance, it would be best to keep everything on separate small drives. The system can read and write to multiple drives faster than to a large single drive.

Database Consistency

Each time you boot your Linux system, the **fsck** program is run against all your file systems. The fsck program makes sure all the directories and data are consistent. It is possible for a file system to become out of sync, causing a loss of data.

The Sybase database server has a similar command. It is called **dbcc** (database consistency check). This command makes sure all data allocations are proper and accounted for. These checks should be run regularly on your databases. They can be scheduled just as you schedule your backups.

The Bookstore Application

You can download the entire application from the *Linux Journal* FTP server at <ftp://ftp.linuxjournal.com/pub/lj/listings/issue64/>. To save space, the entire application isn't presented here. There are two **tar** files: 3250src.tgz and 3250cgi.tgz. The first one has the source code for all database objects. Log in as the sybase user and unpack the tar files. Read the README file before installing. You'll need to install the pubs2 example database shipped with the server. The second tar file has all CGI scripts and other files needed for the application. These files should be placed in the /cgi-bin directory of your web server. Again, read the README file before installing. Sybase and Sybperl must be installed to use this application. Information on this installation is in the first two parts of this article.

This isn't an article about web development, so I won't go into great detail about the web aspects of the application. It also isn't a web application beauty contest. You will see I didn't spend much time making it look nice. I'm sure you can find the time to make the application look the way you like it.

If you installed the application in the default location, you can start a web browser on your system and go to <http://localhost/cgi-bin/store/store/> to run the application. If you receive a "File Not Found" error message, your CGI application cannot be found. On a default Red Hat 5.x system, the CGI perl script should be located at /home/httpd/cgi-bin/store/store.

If you received an Internal Server Error message, your Perl program probably isn't located at /usr/bin/perl. Edit the file /home/httpd/cgi-bin/store/store/ and modify the path for Perl to point to your Perl executable. One way to figure out the problem is to run the Perl script at the command line. You should receive HTML. If you receive an error message, you'll need to resolve this problem.

If you receive the store menu, click on Search for Books. If you receive an error message, your client cannot connect to the database. Look at the file /home/httpd/cgi-bin/store/store.inc/. You may need to change the name of the database server. Make sure your database server is up and running. You should be able to connect to it using the user name and password in the store.inc file. Use the **isql** program at the command line to test this.

Each screen of the application has a subroutine written in Perl. The subroutine is named after the last part of the URL. To search for books, the URL would be <http://localhost/cgi-bin/store/store/search/>. The subroutine responsible for this is called **search**.

The Application Database

In our application, we use two databases. The first is the pubs2 example database. We use the titles table as the list of books. This table is read-only. We just view it; the application does not make changes to it. The second database is called book_d and it is created on two devices. The log segment is on device02 and has 10MB allocated to it. The data segment is on device01 and has 20MB allocated to it. There are five tables in this database.

- **inventory_t**: this table is a one-to-one relationship to the titles table in the pubs2 database. There is one row in this table for each row in the titles table. This table contains the number of books in stock and the number on order.
- **orders_t**: this table is a list of customer orders.
- **order_nbr_t**: this table has one row in it. It is used to guarantee that the order number is always unique.
- **types_t**: this table is a list of the types of books. It is used to populate the drop-down box on the search screen.
- **user_t**: this table is a list of customers.

To provide access to the data, we use stored procedures in all cases except one. Stored procedures are SQL code stored in the database. This allows us to encapsulate procedures in the database so it doesn't have to be replicated in all your applications. It also provides a performance benefit. When the stored procedure is loaded, the SQL is precompiled. When the application runs, the server doesn't have to precompile the SQL, so the application should run faster.

We used SQL in the search1 procedure, because it would change dramatically depending on the parameters given.

In the scripts, we have also created a user to access the database. Each time the CGI script runs, it logs in as this user. The end users of your application do not need to know this, and in fact, shouldn't know it. Even though this user has the minimum rights necessary to run the application, you should protect this user name and password.

This is a very small application. A few more things must be done if you put this application into production.

- Make it look nicer. This application would have looked fine in the early 90's, but in today's world, it would need to look much better before people would use it.
- Prepare and schedule automatic backups. Use **cron** to back up your data at regular intervals.
- Prepare and schedule database consistency checks. These should be run regularly, again using cron to schedule these checks.
- Create and tune table indexes. These tables have no defined indexes. An index is extra data about the table that will allow the database server to access the data faster.

Explaining how the database server uses indexes is beyond the scope of this article—I have a complete book on the subject. Before you put an application into production, you should load example data in all the tables and test queries to be run in your application. Based on the queries, you can make good guesses as to which indexes will be needed. There is a command in SQL that can help tune indexes. It is **set showplan on**. Entire book chapters are devoted to explaining this command and its output.

Summary

As you can see, using a Sybase database as the basis for your client-server applications is not a trivial task. The Sybase database server is an industrial-strength database, capable of handling hundreds of users and many gigabytes of data. Many databases are available for Linux. Of the free ones, I believe the Sybase database server is the most powerful. If you don't need a powerful database, there may be better choices. If, however, you need a high-power database to manage a large amount of data or many users, a Sybase database server would be a solid foundation.

Resources

Jay Sissom (jsissom@indiana.edu) is responsible for the web-based front end to the financial decision support data at Indiana University. He has installed and supported Sybase databases on many operating systems and has written database clients for the web. When not programming, he enjoys amateur radio and playing his bass guitar and keyboards.

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

[Advanced search](#)

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

[Advanced search](#)

Plug-and-Play Hardware

David Cantrell

Issue #64, August 1999

How to make those pesky new PnP sound cards work and play well on your Linux box.

Chances are, you or someone you know has had to deal with plug-and-play (PnP) hardware. These devices are just like “legacy” devices, except they have no jumpers to configure the resources (i.e., IRQs, DMAs and I/O addresses). PnP cards expect the computer, either the BIOS or the operating system, to find and configure the card. It's an excellent idea, but why?

Most Linux users enjoy configuring their computers—getting their hands dirty working with the hardware. On the other side of the coin is the user who expects his computer to turn on in the morning and be ready for him to complete his work without intervention by him. This type of user will also add hardware to his computer, but does not want to spend time finding free resources and setting them. PnP hardware is ideal for him.

This user can purchase a plug-and-play device, put it in his computer, and the configuration is done automatically. This idea of automatic configuration was designed into the PCI bus, the newer of the common busses. Unfortunately, most users still rely on the standard architecture defined by the industry; that is, the ISA bus. It works well, but still has its limitations. Most notably, plug-and-play was not considered when ISA was designed.

The designers of the plug-and-play standard decided to extend plug-and-play to ISA devices. Wouldn't it be nice to have some sort of detection and configuration scheme to find these devices and set them up for the user? Think how easy computers would be! So, the designers set up a scheme to find ISA cards and configure them to the jumpered resources on the card. Hardware vendors loved the idea and decided to cooperate by suggesting the removal of all the jumpers.

Now we have a fully functional plug-and-play system for the ISA bus, right? Well, not exactly. It still has a way to go before it is all the way out the door. PCI architecture will take over in the next few years before plug-and-play ISA works without any problems. For now, we have to work with these plug-and-play ISA devices while mourning the loss of our beloved jumpers.

Using Plug-and-Play Devices

PnP ISA devices can be used in several ways under Linux. The kernel can find and configure them before loading any other drivers. This method sometimes fails to find all devices and to initialize the devices. It also requires patching the kernel with one of the PnP add-ons.

Another way, which I recommend, is to use an initialization program at the user level. You create a configuration file with your devices and specify the resources each device is to use. Then, this file is read by an initialization utility that configures the devices. However, this method does require the drivers for these devices to be compiled as kernel modules.

I have tried a couple of these user-level configuration programs, and the one I have found to be the easiest and most reliable is the **isapnptools** package. This program is designed to work on systems with or without a PnP-compatible BIOS. It will not interfere with "legacy" ISA devices either (devices with jumpers); at least, I have not had that problem.

As an example, I will configure a PnP sound card and load the driver for it. I will assume you can install the card and you have a working knowledge of Linux. Here's a basic overview of the setup process:

- getting the **isapnptools** package
- dumping all possible values
- choosing your resources
- testing the configuration
- compiling the kernel driver
- enabling the device at bootup

Getting the isapnptools Package

Most of the latest Linux distributions come with the **isapnptools** package. However, you may want to grab the latest version from the **isapnptools** web page, <http://www.roestock.demon.co.uk/isapnptools/>. At the time of this writing, the latest version is 1.17.

After downloading the file, extract the contents and compile it. Some systems may require editing the Makefile. Check the included INSTALL file for more information. As root, I used the following commands to compile and install the package:

```
tar xvzf isapnptools-1.17.tar.gz
cd isapnptools-1.17
make
make install (as root)
```

The installation will create the **isapnp** and **pnpdump** programs along with their accompanying man pages and sample configuration file.

Dumping All Possible Values

After you've compiled and installed **isapnptools**, it is time to configure your devices. This is done by creating a configuration file explaining the device(s) and which resources it will use. The great thing about **isapnptools** is that it will build this file for you. Then you go in and play "multiple choice" (as the author puts it).

To create the configuration file, run the **pnpdump** program. It scans for all ISA plug-and-play devices and all possible configurations. These values are dumped to standard out (STDOUT) in the file format needed. Here is the command I use to create the file as root:

```
/sbin/pnpdump > /etc/isapnp.conf
```

Listing 1. Configuration File Start

The program failed on me a few times, but that was mainly due to other system errors. You should now have a configuration file in /etc if you used my command. The beginning of the file should look something like Listing 1. The rest of the configuration file will contain sections for the different devices it found. Now you need to choose which resources the device will use.

Choosing Your Resources

The configuration file may seem a bit confusing at first, but it needs only a little deciphering. The basic layout for a device section is shown in Listing 2. By default, a listing of all possible resources for that device will be within the device section. Basically, just uncomment the lines for the resources the device will use. The lines are commented with # marks.

Listing 2. Device Section

Editing this file requires knowing a bit about the device to be configured. Typically, the user manual for the device will list which resources are required to use the device. In my case, with the Sound Blaster AWE32, I needed a Base I/O address, one IRQ, an 8-bit DMA channel, a 16-bit DMA channel and a MIDI synthesizer I/O address. Other resources are on my card, but I am not using them in this example.

The example device section in Listing 2 shows the resources I chose for my card. I chose 0x0220 for the Base I/O address, 5 for the IRQ channel, 1 for the 8-bit DMA channel, 5 for the 16-bit DMA channel and 0x0330 for the MIDI I/O.

Note that the I/O addresses are called IO 0 and IO 1, and the DMA channels are called DMA 0 and DMA 1. This may make it a little difficult to map the right values to the 8-bit DMA and 16-bit DMA. However, if you read your configuration file after running **pnpdump** and look at the default resource settings in the user manual for your device, you can easily match things up.

In my case, I know DMA 0 corresponds to the 8-bit DMA because only the 8-bit DMA can have a value of 1. So, the other setting must be the 16-bit DMA channel. The same goes for the IO settings; IO 0 must be the Base I/O address, because the MIDI I/O address can never be 0x0220.

Once you have uncommented the lines you need, make sure the **(ACT Y)** line is uncommented; otherwise, your device will not be configured.

Testing the Configuration

You have now passed the hard part of PnP configuration under Linux. It is a good idea to test your PnP configuration before going any further. You want to make sure **isapnp** can properly initialize the card with the resources you have set. Assuming **isapnp** is in `/sbin`, execute this command to test your configuration:

```
/sbin/isapnp /etc/isapnp.conf
```

If there are no error messages, your configuration should work fine. If you do get resource conflict errors, now would be a good time to go back and edit that configuration file. It is better to get it working now than have to fool with it later. Play around with the resource settings until you find those that don't produce errors when you test the configuration.

Compiling the Kernel Driver

Now we are ready for the fun part. Most Linux distributions will come with the various device drivers compiled as modules. Refer to the kernel documentation

for more information about modules. Basically, a module is just a device driver that can be added to an already-running kernel. This provides a lot of flexibility for the user, and it is modules that allow us to use PnP devices.

PnP devices must be initialized before the driver can be loaded, so using modules is a necessity. They are the only part of the kernel that can be loaded after the kernel boots.

Your distribution may already include a module for the device you want to use. In the case of sound cards, you might be compiling one from scratch. In my example, I use a Sound Blaster AWE32. The device driver included with my distribution is `/lib/modules/2.0.35/misc/sound.o`.

If you must recompile, be sure to set any resources for the driver according to those set in the `isapnp.conf` file. You can always pass the resource values when loading the module, but having the default ones is always nice.

To load the driver for my PnP sound card, I did this:

```
/sbin/isapnp /etc/isapnp.conf
/sbin/modprobe sound.o io=0x0220 irq=5 dma=1\
dma1=5
```

If all goes well, you should have a driver loaded and working with that device. Check to make sure the module is loaded by typing:

```
/sbin/lsmmod
```

If this is a sound card, try playing sound files. For network cards, try bringing up the device with **ifconfig**. At this point the device should be working.

Enabling the Device at Bootup

Now that you have your PnP card working under Linux, wouldn't it be great if those devices were set up automatically at boot time? I added the above two lines to my `/etc/rc.d/rc.local` file, so that my sound driver is loaded each time I boot the system.

This example used what I view as the most common plug-and-play situation when using Linux—a sound card. Other PnP devices exist, such as modems and network cards. The same technique can be applied to those cards as well.

Resources



David Cantrell is a Computer Science undergraduate at the Georgia Institute of Technology. He has been using Linux since 1997. David spends a lot of time at his computer, but also enjoys visiting his favorite Mexican restaurant and midnight movies. He enjoys water skiing, backpacking and watching “The Simpsons”. He can be reached via e-mail at david@burdell.org.

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

[Advanced search](#)

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

[Advanced search](#)

Open Source Remote Sensing Effort

Dr. . Shawana P. Johnson

Issue #64, August 1999

Remote sensing software is being developed using the Open Source model by the web project at remotesensing.org.

Wouldn't you like to fly through space like Peter Pan and view the earth from above, explore another country, and find out what exotic places are like while never leaving your own home? That is what satellite and aerial images from space can provide for you. The technology exists to do this, but it would take thousands of technologists to develop the programming capabilities for the average global consumer to access such a "View of the World". This has long been a quest for governments and researchers around the globe. The problem is satellite and aerial images are geospatial data and contain more raw data and information than can be easily dealt with.

The Fascination with the Earth from Space

Everyone is fascinated with images from space which provide close up views of earth. Whether the image is used to study the commodities market for agriculture or provide outstanding graphics to help us catch the bad guy in a spy novel or a stunning backdrop for a James Bond movie, we all want to see more. Today's satellite and aerial technology allows us to see the earth from space with increasing detail. Typically, scientific applications of this earth imagery are making their way into the commercial world through real estate offices, investment banks, flight simulators and even the placement of wireless signal towers. Several new high-resolution satellites will be added this year to provide images where one pixel will represent one square meter of the earth's surface. Earlier satellites provided only 28- and 10-meter resolution. This much resolution means even more data to manage.

Technology Brings Space to Our Industry

How can the average consumer take advantage of this exotic information from space? Until now, he couldn't, but ImageLinks, Inc., of Melbourne, Florida, a division of AGIS, is changing all this with a web site for open-source development of remote-sensing software. Mark Lucas, the chief technical officer of ImageLinks, Inc. struck a harmonious chord among programmers worldwide when he opened remotesensing.org. Remote sensing is the term used to refer to images collected by remote cameras and other sensors from space. These cameras are located on satellites and high-altitude airplanes.

Only with the advent of the Linux operating system could an open-source project like this be considered. Linux provides the software for porting applications and a platform for development. Without Linux, Mark Lucas might not have been able to propose the adoption of the open-source development model for the remote-sensing software industry.

Open Source Development Brings Space to Your Doorstep

remotesensing.org is currently consolidating existing open-source remote sensing and GIS tools as well as developing new libraries and applications. Currently, three separate development efforts are at the site:

LIMP, the large image manipulation program, is headed up by Valient Gough and will evolve into the centerpiece for subsequent image processing applications. Val is the author of OrthoVista and works for StellaCore Corporation.

GeoTIFF is led by Frank Warmerdam for the development of the leading commercial standard for geospatial tiff files. Frank was the chief architect for PCI's ImageWorks, GCPWorks, GeoGateway technology and most of the existing GeoTiff code. He works as a contract developer.

DEMtools is led by Brian Maddox and is a collection of conversion programs, libraries and tools for handling the topographic elevation data sets used in remote sensing. Brian works as a computer scientist for the USGS Mid-Continent Mapping Center.

With over 250 exceptional programming contributors and business specialists, Mark Lucas stated:

We have really made a big hit in the remote sensing community. There are thousands of people in this industry who just don't have the resources and technology to develop the software applications and algorithms alone. But with hundreds of us working

together, we will be able to develop the keys to unlock the data to make it easy to access and view by the average (or unscientific) global consumer. Whether the consumer just wants to see a picture of his own backyard or view the deforestation of the Brazilian rain forest or the B-9 iceberg near the South Pole, he will be able to do this, and it won't cost him thousands of dollars for the software.

Open-source software development enables the development of software algorithms without spending thousands of dollars for commercial applications. Open-source software development speeds up the development time, produces high-quality code and allows for the maintenance and distribution of the software to be completed by users from around the world on the Net. The tools and source available at remotesensing.org will be released under an open-source license and the absolute beauty of this is that it's free.

Also, remotesensing.org provides a place for open communication between academics, government organizations, entrepreneurial developers and the business community. It has long been a problem to establish communication between these diverse groups. remotesensing.org has already begun to bridge these gaps as is demonstrated by the highly influential participants eagerly sharing their ideas, along with valuable code. This prestigious group of participants consists of NASA engineers, well-known remote-sensing business specialists, developers from well-known software companies and progressive student programmers at leading universities around the world.

The site already has a well-established program for development. ImageLinks, Inc. created and hosts remotesensing.org. Current work includes:

- Collecting changes
- Helping authors synchronize their work
- Operating discussion forums (mailing lists and newsgroups)
- Coordinating bug lists
- Keeping track of and publicizing "work in progress"
- Providing "roadmaps" to the code and projects based on the code

The Highway Ahead

What does this mean for the remote-sensing community and the global consumer? An unparalleled opportunity exists to allow the average consumer to see and understand information about our world that otherwise might not be possible for many years. The flood of raw data about our earth does us no good if we cannot turn it into information which provides broad uses to impact our socioeconomic problems in this borderless world.

remotesensing.org is dedicated to providing the world with affordable data and information capabilities to allow us to take better care of the world. What better way to do this than to bring the finest minds together in the Open Source world which Linux has created and allow them to do just that—create?

Acknowledgments



Dr. **Shawana P. Johnson** (shawana@globalinsights.com) is President of Global Marketing Insights, Inc., which provides strategic sales and marketing planning focused on the global remote sensing community, including companies involved in the development and launch of earth observation systems, as well as companies who sell and process the data. She received her Doctorate in Management from Case Western Reserve University where she specialized in technology transfer and global economics.

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

[Advanced search](#)

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

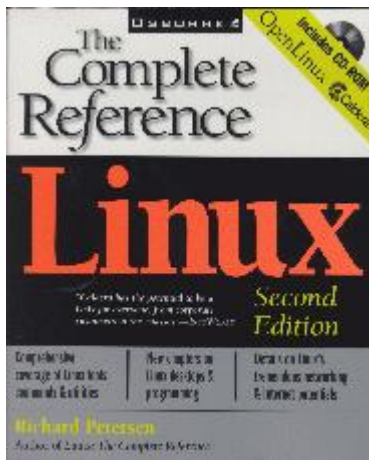
Advanced search

Linux: The Complete Reference, Second Edition

Ben Crowder

Issue #64, August 1999

Even with the seemingly large number of one thousand pages, however, there's a limit to how deep a reader can delve.



- Author: Richard Petersen
- Publisher: Osborne McGraw-Hill
- E-mail: customer_service@mcgraw-hill.com/
- URL: www.osborne.com
- Price: \$50 US
- ISBN: 007 882 461 3
- Reviewer: Ben Crowder

Linux: The Complete Reference, Second Edition attempts to cover the whole of the Linux knowledge base, and in that respect, it makes a fair showing. You may, however, want to wait for a future edition, the reasons for which I'll explain later.

The book is fairly well-organized, granting a wide variety of topics a decent amount of coverage. Even with the seemingly large number of one thousand pages, however, there's a limit to how deep a reader can delve; a few subjects

seem spread a bit too thinly (which is understandable, of course). The book is divided into seven parts: Introduction to Linux, Linux Operations, Networking, Shells, Editors and Utilities, Administration and Appendices. The one noticeably absent area is multimedia—the only mention of anything having to do with video, sound, graphics or gaming is a two-page procedure on installing a sound card. Perhaps multimedia will be dealt with in a future edition. Other than ignoring multimedia, however, the areas chosen seem to cover the geography of Linux fairly well.

Part I begins with an introduction to Linux, describing the history of UNIX and Linux, then gives an overview of the whole system (the shell, the file system structure, utilities, etc.). The next chapter leads the reader through the installation process. This chapter is heavily Caldera-oriented (the book comes with a Caldera OpenLinux Lite v1.2 CD), and you probably won't have much luck trying to install another distribution with these instructions. Chapter three runs through basic Linux tasks such as getting into your system through LILO, logging in and out, starting X Windows, the manual pages, etc. Chapter four tells you about window managers and desktops (such as the Caldera Desktop, with no mention of KDE or GNOME). Later versions of OpenLinux include the KDE desktop, so hopefully this will be updated soon.

Chapter five opens Part II with a guide to shell operations (redirections, pipes, scripts, etc.). Chapter six describes the Linux file structure, and the next chapter goes over file management operations such as permissions and mounting file systems.

In chapter eight, which begins Part III (Networking), one learns how to run the basic e-mail utilities such as Mail, Elm and Pine. Chapter nine goes over Usenet and newsreaders. Chapter ten introduces several Internet tools (TELNET, FTP, archie, gopher, etc.), and the Web is covered in the next chapter. In chapter twelve, the process for creating many different types of Internet servers (web servers, FTP servers, gopher servers, etc.) is described in detail. Chapter thirteen goes over remote access (UUCP, rsh, etc.).

Part IV, Shells, delves into the filters and regular expressions, the Bourne Again Shell (BASH) and the TCSH shell. Here you learn how to write your own shell scripts, as well as the arcane technicalities of each of the two major shells.

Part V covers vi and Emacs in adequate detail. Though the title of the section includes the word “utilities”, only those two editors are covered.

Part VI, Administration, goes into systems configuration (chapter nineteen is mistakenly labeled “Device Configuration” in the header of each page), network administration (the basics of TCP/IP, PPP and SLIP), X Window System

configuration, typesetting (TeX, LaTeX, Ghostscript), the standard Linux C compilers and libraries (gcc, g++ and gdb), Perl, Tcl/Tk, Expect and gawk.

The last and seventh part is just a compilation of appendices. The first lists hardware parameters that may be necessary to pass to the kernel at boot time, such as those for CD-ROM IRQs and other sometimes-annoying settings that don't properly autodetect. The second lists all the software packages that come with the OpenLinux CD, and the third lists the video cards supported by the X Window System.

Apparently, the book was published before it was thoroughly proofread—it is littered with glaring typographical and grammatical errors, misspellings and repetitions (for example, some text is repeated later on in the same chapter, almost word for word). Linus Torvalds' name is repeatedly spelled without the final "s", and `'/dev/cua1'` is used as a plural. Given extensive proofreading by an editor with a sharp eye, and adding a section on multimedia, the book could become a shining jewel, but in its current state it isn't worth the price of \$50 US.



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](#)

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