

Advanced search

Linux Journal Issue #98/June 2002



Features

Kode KDE Kindly, Kan You? by Jason Mott

Help Linux conquer the desktop with your own KDE app.

Emacs: the Free Software IDE by Charles Curley

Not just for text editing—Emacs is the IDE that's been there all along.

Mediator/Python by Doug Farrell

Sure it's no system for a basis of government, but Python can help build smart dialog boxes.

Interview

Python 2.2 Q&A with Guido van Rossum, Creator of Python by Wesley J. Chun

No full monty, just Guido's honest opinions.

Indepth

The OSCAR Revolution by Richard Ferri

Making clusters easy to build for the nonprogrammer.

Toolbox

Kernel Korner A NATural Progression by David A. Bandel

At the Forge Zope Page Templates by Reuven M. Lerner

Cooking with Linux Programming Life! by Marcel Gagné

Paranoid Penguin [BestCrypt: Cross-Platform Filesystem Encryption](#) *by Mick Bauer*
[GFX Silicon Grail RAYZ](#) *by Robin Rowe*

Columns

Focus on Software [Striking a Nerve](#) *by David A. Bandel*
Focus on Embedded Systems [Embedded Systems Conference 2002](#)
by Rick Lehrbaum
Geek Law [Bad Law](#) *by Lawrence Rosen*
Linux for Suits [Identity from the Inside Out](#) *by Doc Searls*

Reviews

[Hewlett-Packard x4000 Workstation](#) *by Thad Beier*

Departments

[Letters](#)

[upFRONT](#)

From the Editor [From the Editor](#) *by Richard Vernon*

[Best of Technical Support](#)

[New Products](#)

[Archive Index](#)

[Advanced search](#)

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

[Advanced search](#)

Kode KDE Kindly, Kan You?

Jason Mott

Issue #98, June 2002

With a little programming experience, you'll have everything you need to build applications for the Linux desktop when you're done reading this.

Kode KDE Kindly, Kan You?

With a little programming experience, you'll have everything you need to build applications for the Linux desktop when you're done reading this.

by Jason Mott

There are many toolkits to choose from for building Linux desktop applications. Some say this is Linux's downfall; others say it is its greatest feature. I'll stand somewhere in the middle and say choice is good if you choose what meets your needs. Most graphical user interfaces (GUIs) on Linux are based on X, a client/server architecture that allows for networked computers to share GUI applications. With X, the application is the client that sends its graphical output to an X server. The X server accepts applications' output on behalf of its local hardware (or sometimes virtual hardware, but I won't go there in this article). In most cases, the X server and the X client are on the same machine, but use the client/server architecture nonetheless.

The base-level toolkit for building an X client is called Xlib. Xlib is much too low-level and difficult to use by itself to build an application from scratch. As a result, many toolkits have been built on top of Xlib to make it easier to write GUI applications for X. Subsequently, when writing GUI applications using one of the high-level toolkits, you never even know it's a networked application (that it's sending its graphical output to a server).

The two most popular open-source toolkits that layer on top of Xlib are Qt and GTK+, upon which KDE and GNOME are respectively built. Motif is another popular toolkit (not open source, but there is an open-source clone called Lesstif). Figure 1 shows a diagram of these (and more) and their relationship

with one another. The farther down on the diagram, the lower the level of API it is. I prefer KDE/Qt for many reasons, but mostly because it's focused on a good user interface and a clean and well-designed API.

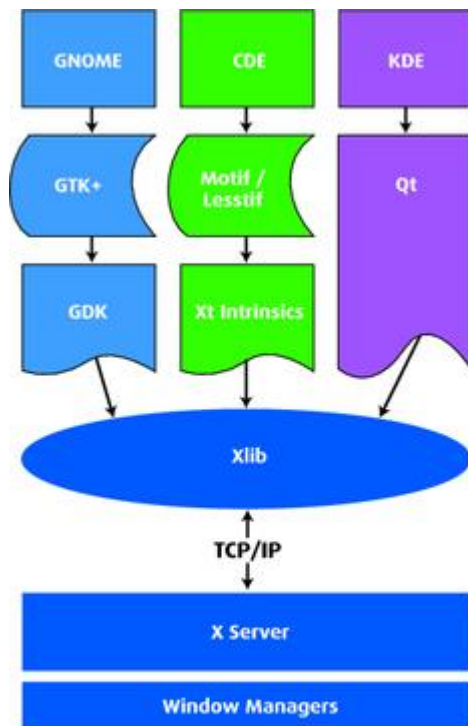


Figure 1. Relationships between KDE and GNOME and Various Toolkits

Requirements

I actually built an application from scratch for this article (a calculator) to show how quick and easy it can be. To follow along you'll need some tools—most importantly, KDevelop. I used version 2.0.2 running on KDE 2.2.2. When using KDevelop's setup wizard you'll be told what dependencies you're missing. Visit the KDevelop web site (www.kdevelop.org) or check with your distribution's included software, and get KDevelop installed. If you have KDE as part of your distribution, then you'll have KDevelop.

KDevelop

Most tutorials on application development try to be IDE-agnostic. I've decided against this for a few reasons: KDevelop is free and comes with KDE; KDevelop makes an already easy API even easier, and KDevelop's wizards help the developer conform to KDE user-interface standards. For those who don't like to deal with Makefiles or creating configure files (or even dealing with the easier automake and autoconf), KDevelop will handle these things for you. And for geeks who do like to mess around with Makefiles, KDevelop allows you to tweak them. In fact, your KDevelop application is ready to be built with the

traditional **./configure, make, make install**. In other words, you're free to break free from KDevelop anytime you want.

The first time you fire up KDevelop it will run you through a setup wizard. The most important piece of the wizard is when it checks for dependencies (Figure 2). Take the time to look through this output; if there are failures listed, do yourself a favor and try to find the libraries needed. The easiest way to do this is to use www.rpmfind.com (if your distribution is RPM-based). For each library that failed, do a search for it on the rpmfind web site. From the search results, select the RPM appropriate for your distribution and install it. Once you've done that, rerun the KDevelop wizard (type **kdevelop --setup** on the command line or find an entry in your K menu that says KDevelop Setup). If there are still missing items, repeat as necessary.



Figure 2. KDevelop Setup Wizard

Kalculate

The calculator I built is simple and only does basic arithmetic. The whole thing took about two hours to complete (okay, three hours, but I wasn't in a hurry). I was going to name it Kcalculator, but there is an obscure application out there that already uses that name. So I chose Kalculate. I did not use nor view any code from KDE's stock calculator called KCalc, which has more mathematical features than mine. I've always felt KDE should have a basic calculator anyway, so I'd like to see user-friendly features added to Kalculate rather than more mathematical features. Therefore, when you're done reading this you can be a part of a living experiment in open-source development—come join the team and add a feature! You'll get special preference if you say this article sent you (sourceforge.net/projects/kalculate).

The Document-View Model

We are not going to start with a new project, but rather with the power of the Internet; we're going to use Kalkulate. To get the code for Kalkulate from the official CVS server, type the following (assuming you have write permissions, create a directory called `/usr/local/src` if you don't already have one):

```
cd /usr/local/src
cvs -d:pserver:anonymous@cvs.kalkulate.sourceforge.net:
/cvsroot/kalkulate login
```

If this command asks for a password, press enter; no password is needed.

```
cvs -z3 -d:pserver:anonymous@cvs.kalkulate.
sourceforge.net:/cvsroot/kalkulate co kalkulate
```

Of course, if you want to build this application yourself, you wouldn't grab the source. But because I'm not going to do a line-by-line rundown here, it's probably best to get the source and then use your new knowledge to either add features or build a new application.

The design pattern used here is called the Document-View model. It starts out with three application-specific classes: `KalkulateApp`, `KalkulateDoc` and `KalkulateView`. These three classes would have been created for you if you were freshly starting this project with KDevelop. For Kalkulate, I did not add any additional classes other than what is provided in the template. I did this on purpose so that it would be easy to demonstrate the design pattern and to show just how far along KDevelop starts you. All I had to do to build a fully functional calculator was to modify three classes, add a couple of icons and *voilà!*

The basic rundown of Document-View is that the business logic goes into the document end, and the user-interface stuff goes into the view end. In theory, there could be several views of the same document. In the generic sense, a "document" is one open session of a "thing". Naturally, this is easiest to comprehend with something like a text editor. In that scenario, the document object would represent all the behaviors and properties of a text file, whereas the view object would provide the display and user interface of the text to the screen. You'll even notice in the skeleton code that saving, printing, opening and closing routines are stubbed out for you. Clearly the text editor metaphor is the basis for this model.

However, as is the case for Kalkulate, this metaphor doesn't always work. A calculator doesn't really have a file it represents; it has a certain set of functionality it represents. So although the `KalkulateDoc` class will provide our business logic, we have no need for the open, close, save and print functionality. For now, in the Kalkulate code, I commented out the stock code

(that KDevelop provides) that I didn't need instead of deleting it. This way, if the need arises, we still have the stub.

What we do need for a calculator, on the business logic side of things (CalculateDoc), is to be able enter numbers, assign actions for those numbers and get the resulting number (and of course, internally it actually needs to perform the actions). On the user-interface side, we need buttons for entering the numbers and entering actions, and we need a display of the numbers. Assuming you put the cvs checkout into /usr/local/src, you'll have a /usr/local/src/kalculat directory. Type the following:

```
cd /usr/local/src/kalculat
kdevelop kalkulat.kdevprj &
```

This will open up the project. KDevelop stores its project info into a file named after the project with a .kdevprj ending. Depending on how you have your preferences set up, you should see something like Figure 3. If you don't see a project tree on the left side, make sure that View->Tree Tools Views->Files is selected. You will see the project file structure click into the kalkulat directory (yes, this is a kalkulat directory under a kalkulat directory, don't clear your eyes). Here are all the source files. Of particular interest now are the calculatedoc.cpp and calculatedoc.h files. Double click them. These two files are the core of our calculator—the CalculateDoc class handles all of the processing. It's a virtual machine that needs someone to hook up controls to it. If you view calculatedoc.h, you'll notice something very odd about the class declaration—there's some seemingly incorrect syntax. This brings us to the world of slots and signals.

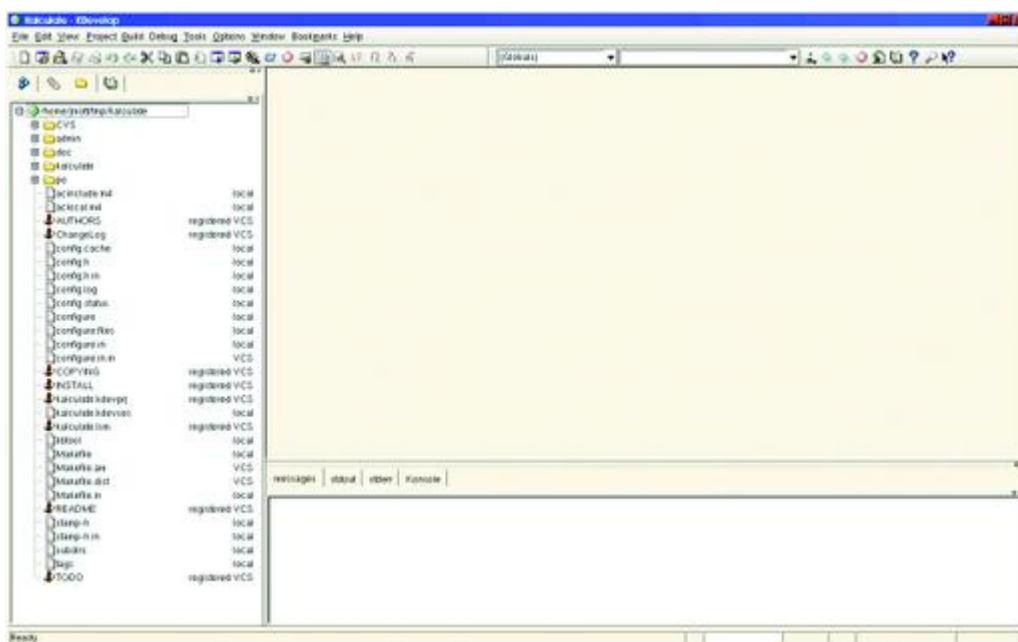


Figure 3. KDevelop Project File

Slots and Signals

In event-driven applications, such as desktop applications, a subsystem needs to be built that allows components to be notified when an event occurs. An event could be some type of user interaction (moving the mouse pointer, pressing the mouse button, pressing a key, etc.). Events also may be other software components announcing some condition (e.g., the time, a full filesystem, etc.). A popular event in GUI applications, and especially Kalkulate, is that a button widget has been pressed. In KDE land, events are called signals, and the things that handle the events are called slots. As the developer, you can create signals and slots, and you can control which slots are connected to which signals. You can also connect your slots to stock signals (and vice versa) or connect stock signals to stock slots. By stock I mean using a KDE or Qt library class that already has slots and/or signals defined.

KDE (and thus Qt) gets away with the incorrect syntax because it uses the standard C++ preprocessor `#defines` that replace `signals:` with `protected:` and `slots` and `emit` with nothing. The extra syntax is needed by `moc` (which is run before the preprocessor). The Meta Object Compiler (`moc`) is a program that comes with the Qt library and is run on your `QObject` header files in order to create extra methods for your classes that make them usable as a `QObject` subclass. All objects that use slots and signals must be a subclass of `QObject`, and the class definition must have the `Q_OBJECT` macro called (as you can see in the `kalculatedoc.h` file). The `QObject` superclass has a `connect` method, and that is the method you use to connect your slots and signals.

To create signals, declare them under `signals:` in your class declaration. For signals, you don't really have to create a subsequent method. All you have to do is call `emit` on the signal name, and any slot that is connected to it will be called; `moc` actually creates the signal method for you.

Slots, on the other hand, do have to be created. You declare your slots under `public slots`, or `protected slots`, and then define them as normal methods in your `.cpp` file. You can then connect them anytime you want. Like this:

```
connect(this, SIGNAL( mySignal() ),
        this, SLOT( mySlot() ) );
```

Assuming you have defined a signal called `mySignal()` and a slot called `mySlot()` within the same class, the above call to `connect` would bind them together so that any time `emit mySignal()` is called, `mySlot()` will get called. If you wanted to connect your slot to some other signal in some other object, then your first argument would be the instance of the object, and your second argument would be the `SIGNAL()` macro with the name of the signal inside the

parentheses. It's that easy. All the hard stuff (including running moc on your headers) is done by KDevelop.

So (back to our calculator) the `KalculateDoc` class is a series of slots. Each slot represents an action one would take on a calculating machine. The header file declares all the slots as shown in Listing 1. Again, the only thing that makes these methods special is that they are under a “public slots:” label, and thus moc will create some metadata about them. As slots, you still need to define the methods as normal (look at `kalculatedoc.cpp` to see my definitions of these slots). Signals, on the other hand, you do not define; moc does that for you.

Listing 1. Slots

So now all we need is a GUI that uses this class. Enter `KalculateView` (I hope you are actually opening up the source code and checking it out). The `calculateview.h` file defines our class that provides the GUI. Under “protected” you'll see that several layout managers, an LCD display and several buttons are declared. I did add one file to KDevelop's stock template code; it's called `calculatesizes.h`, and here is what is in it:

```
#ifndef KALCULATE_SIZES_H
#define KALCULATE_SIZES_H
#define BUTTON_WIDTH 35
#define BUTTON_HEIGHT 35
#define LAYOUT_SPACING 4
#define MAX_WIDTH (BUTTON_WIDTH * 5)
    + ((LAYOUT_SPACING * 2) * 4)
#define MAX_HEIGHT (BUTTON_HEIGHT * 5)
    + ((LAYOUT_SPACING * 2) * 4)
#endif // SIZES_H
```

Basically, it sets up the size of our calculator. I decided to go with a calculator that isn't resizable, but I wanted to make it easy to change the button sizes. So I define the sizes in this file. All one needs to do is edit this file and recompile to get a calculator with different-sized buttons (perhaps future versions could allow for this on the fly, but I'm hesitant for some reason).

In `KalculateView`, constructor (in `calculateview.cpp`) is where this application comes to life. The call to `setMaximumSize()` is what makes this not resizable. The size policy helps out too, but is just a suggestion. I'll take a moment here and explain layout managers. Basically, every GUI widget has methods to set its geometry (height, width, relative position, etc.). When an application is first set up, and when it's subsequently resized, it wouldn't be very fun if you had to write code that resized or repositioned it. And on the initial setup, without a layout manager, all the buttons in this application would have to be positioned with hard-coded x/y coordinates. That could take hours of time when you wanted to change the size of the calculator. So, instead of hard-coding geometry attributes, we register our widgets with layout managers that follow

certain rules, for example, the QVBoxLayout, which adds widgets in a vertical column. Every time you put a widget in with the addWidget() method like this:

```
outerLayout->addWidget(output,1);
```

it adds the widget just below the previous widget. A QHBoxLayout would add the widget just to the right of the previous one. The second argument is what is called the stretch factor. Basically, the stretch factor decides how much space this widget is going to take in relation to the other widgets added to this layout manager. So if they are all 1, then they will all be the same size (unless factors like setMaximumSize override it). It takes the sum of all the stretch factors and applies a ratio. So if you had two widgets, the first with the stretch factor of 1 and the second with a stretch factor of 2, then the second would be twice as big as the first.

The really cool thing (as exemplified in KalculateView) is that you can add layout managers to layout managers. This allows you to create very complex layouts. Go ahead and play around with the Kalculate code; see what you can create.

So, to run this application, do the following. Select Autoconf and Automake from the Build menu. Then select Configure from the Build menu. When this asks for arguments, type **--prefix=[your base kde dir]**. This will allow you to install the application, which is needed if you want the application's icon to show up. On my machine, which is Mandrake, the base directory for KDE is /usr. Your distro may differ. Then, select Execute from the Build menu, and it will compile and run. You should see something like Figure 4. But the icon in the upper left-hand corner may be missing until you install it. To install, as root type:

```
cd /usr/local/src/kalculate
make install
<center>
```



Figure 4. Kalculate

I hope I have pointed you in the right direction. I have barely scratched the surface of all the things you can do with KDE, but I have shown you how quickly KDevelop allows you to get started. It handles almost all the back-end work for you and gives you an application that compiles on the command line without the presence of KDevelop (this is good for distributing your code). Feel free to e-mail me if you have any questions, and I encourage you to join the Kalkulate team and add some features.



email: jmott@users.sourceforge.net

Jason Mott (jmott@users.sourceforge.net) is an independent software consultant currently on assignment at ElementK (www.elementk.com) in Rochester, New York, helping to build their on-line training site. He is also a part-time Linux consultant, and when he has spare time, he builds Linux desktop applications.

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

[Advanced search](#)

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

Emacs: the Free Software IDE

Charles Curley

Issue #98, June 2002

“Many people waited a long time to have nice IDEs for Linux. While everyone else waited I just used Emacs.” —Thanyne Harbaugh, President, Provo Linux Users Group

Emacs is well known as an editor, but calling Emacs an editor is like calling the *Queen Mary* a boat. The source RPM for Emacs 21.1 is some 20MB—huge for an editor. You don't have to use all of it, but sometimes it's nice to know it's there.

Emacs is heavily customizable, which makes it very flexible. For example, I am writing this article with Emacs and will spell check it with Emacs. “Well”, I hear you saying, “if it's good for writing articles, it can't be very good for developing software.” That's where Emacs' flexibility and customization come in; you can use it for either or for something completely different. Want to see a shrink, play *Tetris* or manipulate dates in the Mayan calendar? Emacs.

Emacs' customization comes in packages, called modes. Major modes set Emacs up to do a particular sort of thing, such as the C mode to edit C and C++ source or the psgml mode (www.lysator.liu.se/projects/about_psgml.html) to edit the documentation after you've written your program. Minor customizations provide added facilities to major modes. Show-paren mode shows matching parentheses, and auto-fill mode allows you to enter filled text without explicitly entering line breaks. Most modes have variables to control their behavior, and you can modify them temporarily or permanently.

Most users store their customizations in `~/.emacs`, called the “dot Emacs” file. For the curious, mine is on the Web at w3.trib.com/~ccurley/emacs.init.html. Others can be found at “the very unofficial .emacs home”, www.dotemacs.de.

This article covers GNU Emacs version 21.1 (www.fsf.org/software/emacs/emacs.html). XEmacs (www.xemacs.org) is also available on most Linux distributions and will probably do everything I describe here.

A Brief Program

What I will do to show off Emacs as an integrated development environment (IDE) is walk through the development of a short C program and show you how to do it. We'll write the infamous Hello World program known and loved by C programmers everywhere.

While this article shows C development, Emacs has support for other languages as well, including C++, assembly, Scheme, Java, Ada, IDL, Makefiles, Lisp (including Emacs Lisp, the language in which much of Emacs is written) and FORTRAN. And that's just what comes with Emacs. Add-on packages abound; there is even one for Forth.

In addition, you can use Emacs for many other types of files. When it comes time to document your program, you can use Emacs' psgml mode to make editing the documentation source much easier. There are modes for HTML and TeX as well.

Directory Manipulation in Emacs

The first thing we do is use Emacs' dired mode to create a directory to put the program in. In dired mode, we type a **+**, and then enter the name of the directory to create in the minibuffer at the bottom of the editor (Figure 1). Press Return and Emacs creates the directory for us.

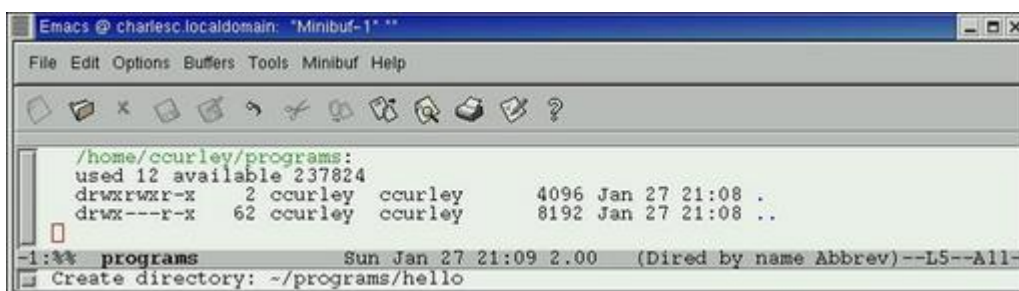


Figure 1. Using Emacs' dired mode to create a directory for the program—we've shrunk the window down to save space in the magazine, but the essentials are there.

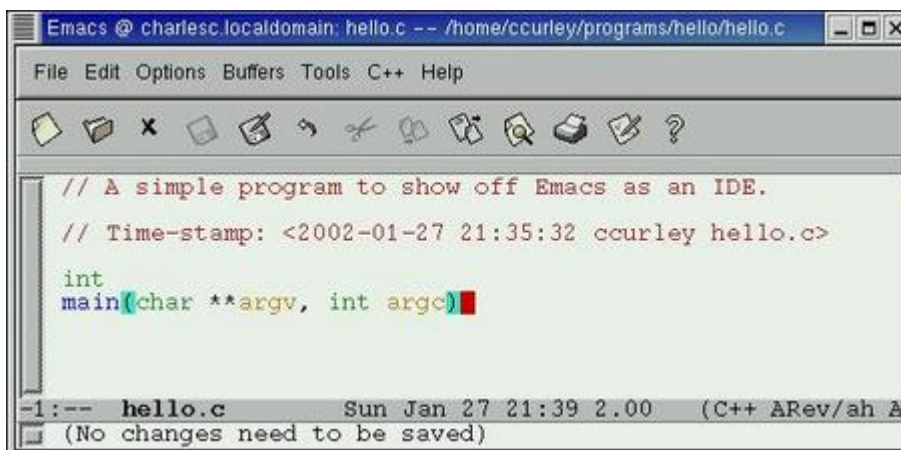
The next thing we do is enter that directory, then create the file `hello.c` with the usual keystrokes for visiting a file, `Ctrl-X Ctrl-F`. Type the filename, **hello.c**, into the minibuffer and press Return. We now see Emacs ready with a new buffer for our program.

Entering the Source

The first thing any C programmer should do upon opening a new file is enter a comment to say what the file does. So we press M-; (M- for meta, or Alt on most PC keyboards) and get a C++-style comment delimiter and the cursor ready to take our comment. The reason I get a C++ comment is because I have customized Emacs to put me into C++ mode for C. You will probably get a C-style comment delimiter, `/* */`, with the cursor in the middle ready for you to type in your comment. If you'd rather use C++ comments, do M-X **C++-mode**.

Then we add a timestamp, all defined by Emacs' timestamp minor mode. Again, this is something I have customized; you may not get it to work.

A C program has to have a main function, and so we type it in GNU style: the return type on one line, followed by the function declaration on the next. Notice as we type away that syntax highlighting takes effect (Figure 2).

A screenshot of the Emacs editor window. The title bar reads "Emacs @ charlesc.localdomain: hello.c -- /home/ccurley/programs/hello/hello.c". The menu bar includes "File Edit Options Buffers Tools C++ Help". The toolbar contains various icons for file operations. The main text area shows the following code with syntax highlighting:

```
// A simple program to show off Emacs as an IDE.
// Time-stamp: <2002-01-27 21:35:32 ccurley hello.c>

int
main(char **argv, int argc)
```

 The status bar at the bottom shows "-1:-- hello.c Sun Jan 27 21:39 2.00 (C++ ARev/ah A (No changes need to be saved)".

Figure 2. Entering the function declaration for main in Emacs. Note the syntax highlighting and also the parentheses matching.

Electric C

Now for one of the really fun things about Emacs: Electric C. Press certain characters, and Emacs takes care of the indentation and the pretty printing for us. Having completed the function declaration, in a normal editor one would press Return, type a left brace (`{`), then press Return and Tab. Not in Emacs; just type a left brace and Emacs does the rest. (If it doesn't work the first time, undo the brace with Ctrl-_, then press Ctrl-C Ctrl-A to toggle electrification.)

Now we type in a line of code. Type in **printf**, then a space. Now it's time for a parenthesis. While we're at it, to keep our parentheses balanced, it's also time for a closed parentheses. So press M-(, that's Alt-(on a PC keyboard. Emacs puts in the pair of parentheses and moves back a character so we can type code inside. And, we type in **"Hello, wolrd.\n"**, typo and all (we'll come back to that).

Instead of pressing Return, we enter another electric character, Ctrl-J. The astute reader has already noticed that we left out a semicolon. Watch what happens. Ctrl-J *should* move us down a line and indent to just below the “p” in “printf”. In this case, it doesn't. Emacs assumes that we are continuing from the line above, so it indents further. Even to a software engineer who has only had his first cup of coffee for the day, this should be an alert that something is wrong. Yes, indeed, we missed our semicolon. We undo the Ctrl-J with Ctrl-Shift-- (that's Control-Shift-Hyphen) and type in the semicolon. Hey! Semicolons are electric, too! But notice how Emacs prevented a bug from happening? That's even cooler!

Now, of course, it's time to enter a right brace in order to end the function main. Because our electric semicolon indented for us, do we have to backspace back to the left column? Nope, just type the right brace.

Compiling and Testing

Now that we have our program written, we save it (Ctrl-X Ctrl-S) and run a test compile. Since this is a single-source file program, we don't need a make file. (But there is a make file mode, which I will leave as an exercise for the student.) Instead, we compile from within Emacs, M-X **compile**. This results in a proposed compile command line, in this case the default **make -k**. Since we don't have a make file, we use the down arrow key and write our own compile command:

```
gcc -g -o hello hello.c
```

The -g option compiles in information for GDB, the GNU Debugger, and the -o option tells the linker that the output file is to be named “hello”. And we get, much to our amazement, a perfect compile (Figure 3).

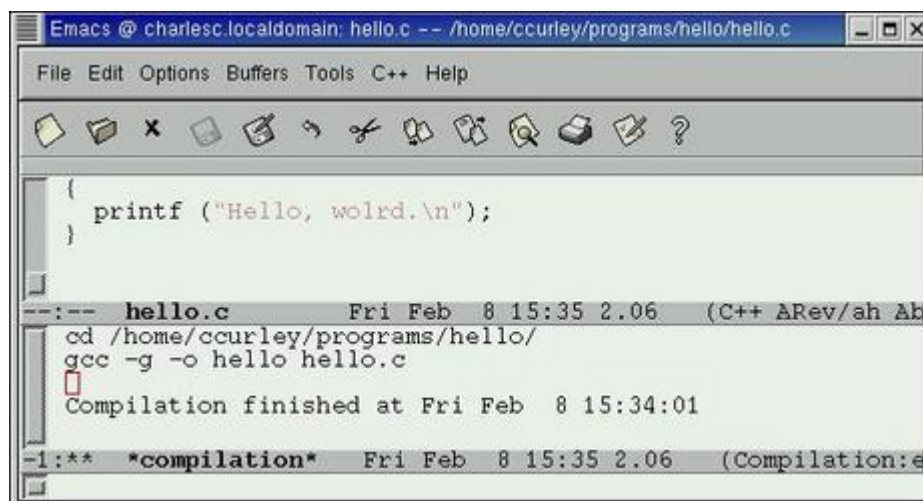


Figure 3. A Successful Compile in Emacs

Now to test it. Feeling brave—or lucky—we fire up a shell under Emacs with M-X **shell** and execute the program (Figure 4). However, something isn't quite right. We misspelled “world”. Oops.

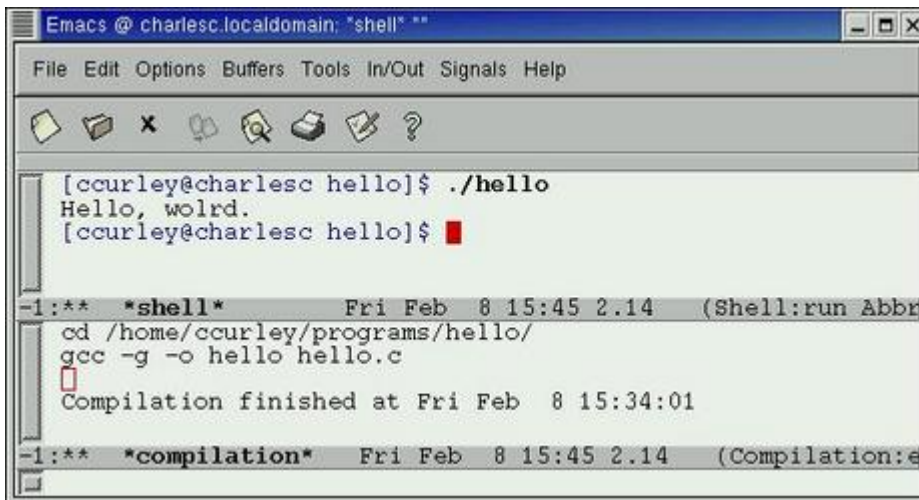


Figure 4. Executing the Program from a Shell inside Emacs

Spell Checking

But rather than just correct that error, we can spell check our comments and strings. To do that, we go back to the source buffer with Ctrl-X Ctrl-B Return. Now either use the pull-down menu (Tools—>Spell-Checking—>Spell-Check Comments) or do M-X **ispell-comments-and-strings** (now is a good time to see the Sidebar on Tab completion). We press the spacebar to bypass IDE and 0 to accept world instead of wolrd (Figure 5). Then Ctrl-X Ctrl-X to save our changes.

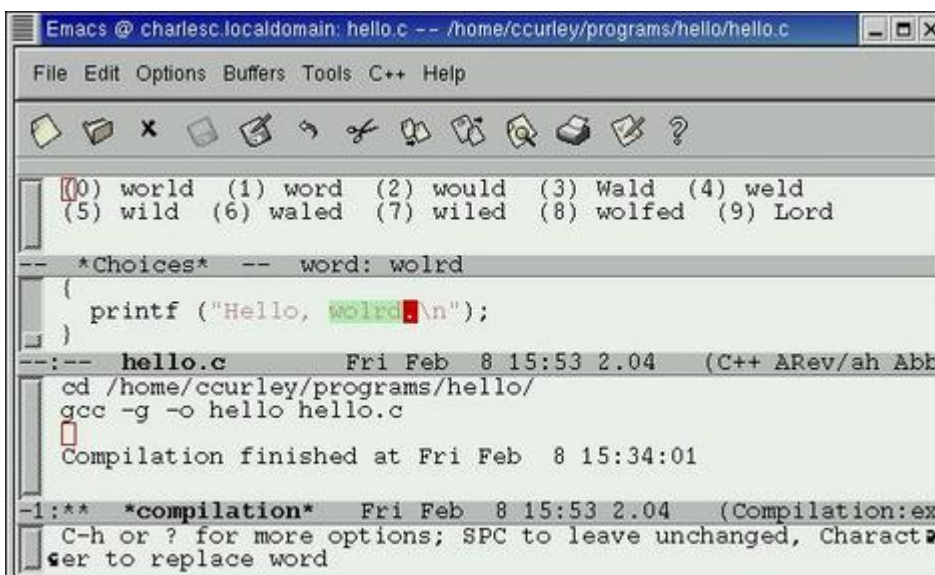


Figure 5. Spell Checking Source Code in Emacs

Now that we have that done, it's time to compile again. Like Bash, Emacs remembers our command history, so M-X followed by one or more up arrow

keys gets us back to the compile command. Press Return. Emacs shows us our last compile command line. It looks good, so press Return again.

Debugging with Emacs

Emacs is also a front end for GDB. To enter GDB mode, do M-X **gdb**. Emacs comes up with a proposed command line, "gdb". We add the name of the executable file to debug, **hello**, and press Return. This tosses us into gdb mode, with the gdb prompt ready for action. We set a breakpoint at the function "main" with the gdb command **break main**. (Tab completion works here, too.) Then type **run** to run the program to our first breakpoint.

GDB shows us the input values to the program. Emacs, however, has the other window set up to show the source. As we step through the program, Emacs will indicate the next line to execute with an arrow in the source window. We can even switch buffers (Ctrl-X Ctrl-O) and edit the source as the fancy strikes. Pressing N executes one statement. In this case, that statement produces output, which we see in the GDB window. One more N and GDB informs us that the program exited with a return value of, well, garbage.

We recall that the successful completion of a program should generally return zero. We should add that to the program. So we quit GDB and switch to the source-code window. The cursor is just in front of the final brace of the program. To add the return code, we enter **return (0)**.

We won't get very fancy in our show-and-tell. But for anything more sophisticated than what we do here, you will find a lot of repetition before getting to the interesting part of the program execution. Therefore, you can prepare an init file for GDB, which is a list of GDB commands that are executed in the order GDB encounters them in the file. So you can automate anything GDB can do—which is quite a lot.

Compilation Debugging

Being in a hurry, we neglected our semicolon and pressed Return instead of Ctrl-J. Try a compile. Do M-X and the up arrow keys until you have "compile" in the minibuffer, then press Return. The command line is good, so press Return again.

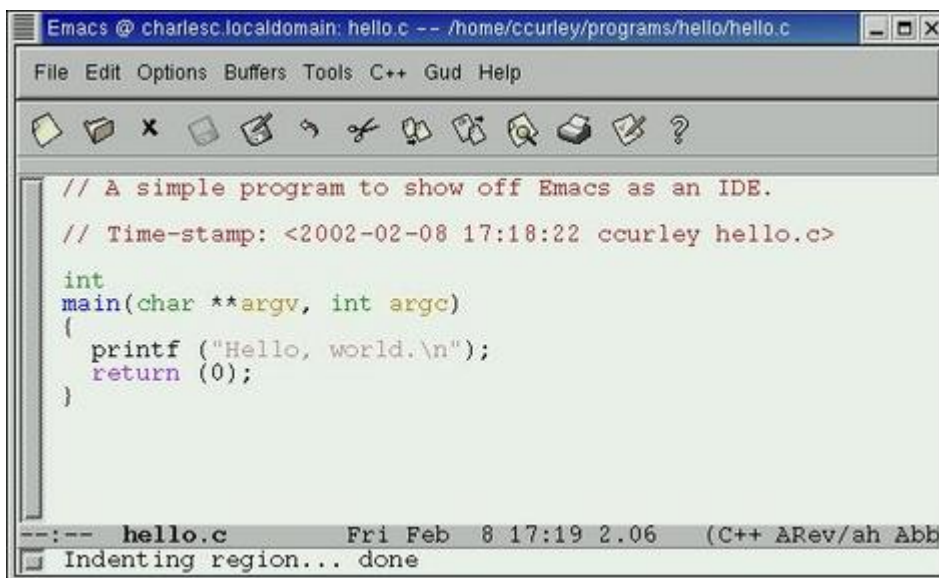
Oops! We have an error. To locate the error in the source, we press Ctrl-X ` . This puts the first error message at the top of the compile output buffer and the cursor at the offending line. The error is before the brace, so we look at the end of the previous line. How terrible; we've committed the classic newbie programmer's error and left out the semicolon. So we press the left arrow key and add the semicolon. Then we save our work with Ctrl-X Ctrl-S.

Emacs' compilation mode is set up to detect error messages from a great many compilers (Microsoft Visual C++, for example) and not only C compilers. Emacs also recognizes the output from weblint, an HTML checker.

Notice that the electric semicolon did two things: it indented our line for us, which was nice of it, but it also inserted a line between the "return" statement and the closing brace, positioning the cursor ready to add another line. Well, that isn't what we wanted. So we undo it with M-Shift--. Now we override the electric semicolon and insert a semicolon with Ctrl-Q ;.

Pretty Printing

It would be nice to have the program all properly indented, so we use Emacs' indentation tools. First we select a region to indent. To select the entire program, Ctrl-X H. Then we indent with Ctrl-M-\ (that's Control meta-backslash). To look at the results, we hide the compilation buffer with Ctrl-X 1 and admire our handiwork (Figure 6). That done, we compile once more (M-X **compile** Return Return). This time we are successful.



The screenshot shows the Emacs editor window titled "Emacs @ charlesc.localdomain: hello.c -- /home/ccurley/programs/hello/hello.c". The menu bar includes "File Edit Options Buffers Tools C++ Gud Help". The toolbar contains various icons for file operations and editing. The main text area displays a C++ program with the following code:

```
// A simple program to show off Emacs as an IDE.
// Time-stamp: <2002-02-08 17:18:22 ccurley hello.c>

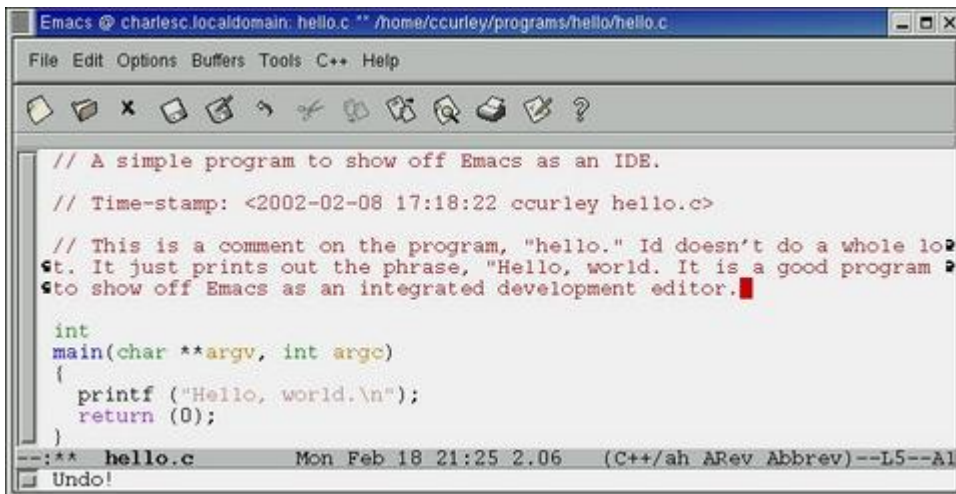
int
main(char **argv, int argc)
{
    printf ("Hello, world.\n");
    return (0);
}
```

The status bar at the bottom shows "--:-- hello.c Fri Feb 8 17:19 2.06 (C++ ARev/ah Abb)" and a message "Indenting region... done".

Figure 6. Pretty Printing with Emacs

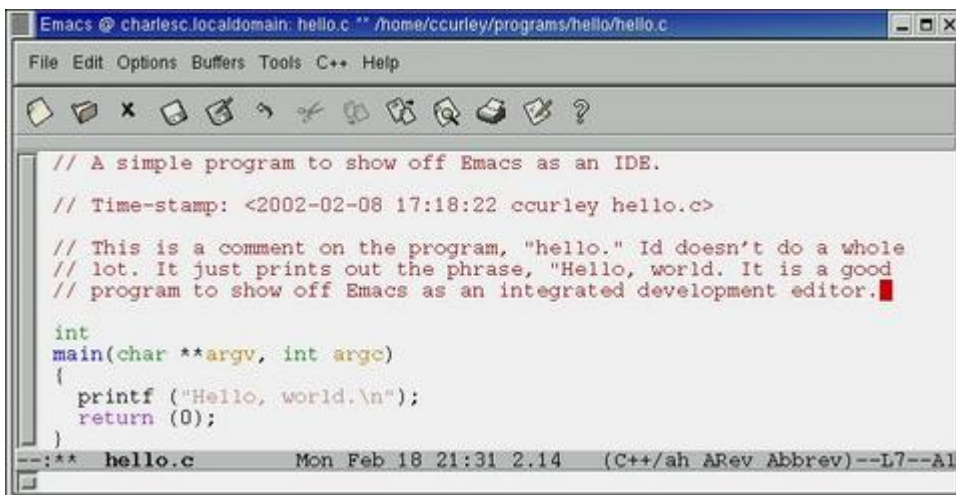
Electric Comments and Auto-Fill Mode

Of course, any programmer knows to comment their code copiously. First, we need a general comment to tell what the program does. So we enter that in the source file right after the timestamp (Figure 7). Start with Emacs' command to start a comment, M-;. Notice as you type that the text simply wraps around (as indicated by the arrows in the left and right margins). That's ugly, so we press M-Q to "fill" the paragraph. Notice that the C++ comment delimiters are inserted at the beginning of each line (Figure 8).



```
Emacs @ charlesc.localdomain: hello.c ** /home/ccurley/programs/hello/hello.c
File Edit Options Buffers Tools C++ Help
// A simple program to show off Emacs as an IDE.
// Time-stamp: <2002-02-08 17:18:22 ccurley hello.c>
// This is a comment on the program, "hello." Id doesn't do a whole lot
// t. It just prints out the phrase, "Hello, world. It is a good program
// to show off Emacs as an integrated development editor.
int
main(char **argv, int argc)
{
    printf ("Hello, world.\n");
    return (0);
}
--:** hello.c      Mon Feb 18 21:25 2.06  (C++/ah ARev Abbrev)--L5--A1
Undo!
```

Figure 7. Entering a Comment (or Other Text) into Emacs



```
Emacs @ charlesc.localdomain: hello.c ** /home/ccurley/programs/hello/hello.c
File Edit Options Buffers Tools C++ Help
// A simple program to show off Emacs as an IDE.
// Time-stamp: <2002-02-08 17:18:22 ccurley hello.c>
// This is a comment on the program, "hello." Id doesn't do a whole
// lot. It just prints out the phrase, "Hello, world. It is a good
// program to show off Emacs as an integrated development editor.
int
main(char **argv, int argc)
{
    printf ("Hello, world.\n");
    return (0);
}
--:** hello.c      Mon Feb 18 21:31 2.14  (C++/ah ARev Abbrev)--L7--A1
```

Figure 8. The Same Comment, Now "Filled"

We can arrange to fill text automatically with M-X **auto-fill-mode**, but this will fill our C source as well, which is not what we want. If you like, you can turn auto-fill mode on for comments and off again for source.

You can also comment a line of code. Put the cursor anywhere on the line, press M-; and continue. Emacs will put in the comment delimiter at the end of the line and move the cursor to the appropriate place to enter the comment. That will be the end of the line for C++-style comments, but between the delimiters for C-style comments, /* ... */.

And, don't forget to spell check your comments with M-X **ispell-comments-and-strings**.

Version Control with Emacs

Emacs has a front end for CVS, RCS or SCCS, called VC. The first two are free-software version control systems and come with many Linux distributions. VC is smart enough to figure out which one you are using. Most VC functions are

handled with the key sequence Ctrl-X Ctrl-Q or Ctrl-X V V. Depending on the present state of the file, VC will check it in or out. If the file has never been added to the version control system, Emacs will determine that and check it in for you. When you check in a change, Emacs will make a temporary buffer for the change comment, so you have no excuse for ignoring that good programming habit.

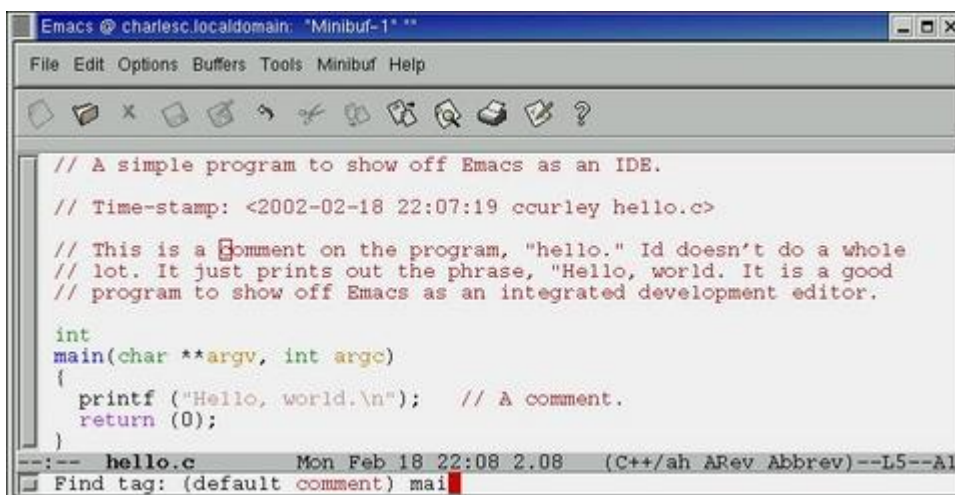
This front end works with any document under version control. For example, the Linux Documentation Project uses remote CVS to control their linuxdoc and docbook source files. Even though I contribute over a 56KBps dial-up line, I can use Emacs to make the CVS process transparent.

Tags

Tags are a database of function, and optionally, typedef names are created with the program `etags` and stored in the file `TAGS`. They are stored together with their locations, making it easy to examine the definition of a function. To create a tags file for C programs, run `etags -t *.ch` from any shell. You can use tags for many programming languages; run `etags --help` from a shell for a list.

Our minimal program has only one file and only one function. Other developers should be so lucky; many C projects spread out across multiple files in multiple directories. Tags will cover complex situations like that as well. In fact, the more complex the project, the better tags shine.

To examine the source for a function, use M-. (that's meta-period). Start typing the name of the function and use tab completion. If you haven't already specified a tags file to use, Emacs will ask. Usually the default is exactly what you want (Figure 9).

The image shows a screenshot of the Emacs editor window. The title bar reads "Emacs @ charlesc.localdomain: 'Minibuf-1'". The menu bar includes "File Edit Options Buffers Tools Minibuf Help". Below the menu bar is a toolbar with various icons. The main text area contains C code with comments. The code is as follows:

```
// A simple program to show off Emacs as an IDE.
// Time-stamp: <2002-02-18 22:07:19 ccourley hello.c>
// This is a comment on the program, "hello." Id doesn't do a whole
// lot. It just prints out the phrase, "Hello, world. It is a good
// program to show off Emacs as an integrated development editor.

int
main(char **argv, int argc)
{
    printf ("Hello, world.\n"); // A comment.
    return (0);
}
```

At the bottom of the window, a status bar shows "--:-- hello.c Mon Feb 18 22:08 2.08 (C++/ah ARev Abbrev)--L5--A1". Below the status bar, a prompt "Find tag: (default comment) mai" is visible with a cursor at the end.

Figure 9. Using the Tags Facility to Locate a Function

Notice that the default function name is the word next to the cursor in the source window. Tags mode lets you edit one file, place the cursor over the name of a function to examine and go edit that function with minimal keystrokes.

You can also display the source for the function in another window, with `Ctrl-X 4 ..`. This lets you examine multiple functions simultaneously, as many functions as you can fit on the screen. If you want to see every reference to a function, use `M-X tags-search`. To continue the search, use `M-,`.

Also, tags are very useful for searching and replacing function names within projects. For example, if I have a function named Frodo and I want to rename it Gollum, I not only need to change the function declaration, but I have to get every prototype and every reference. So `M-X tags-query-replace` and away I go. Both of these search functions use regular expressions, making them very powerful. You can browse C++ classes with Ebrowse.

File Diffing

Diffing is comparing two files, say two versions of a source file. Most programmers are familiar with diff and patch. Emacs provides a powerful front end for diff and patch. For one thing, Emacs' ediff mode shows the two (or three) files in the editor one above the other. The differences are highlighted on any display capable of color (Figure 10). The entire line is indicated, and the exact differences are indicated in different colors.

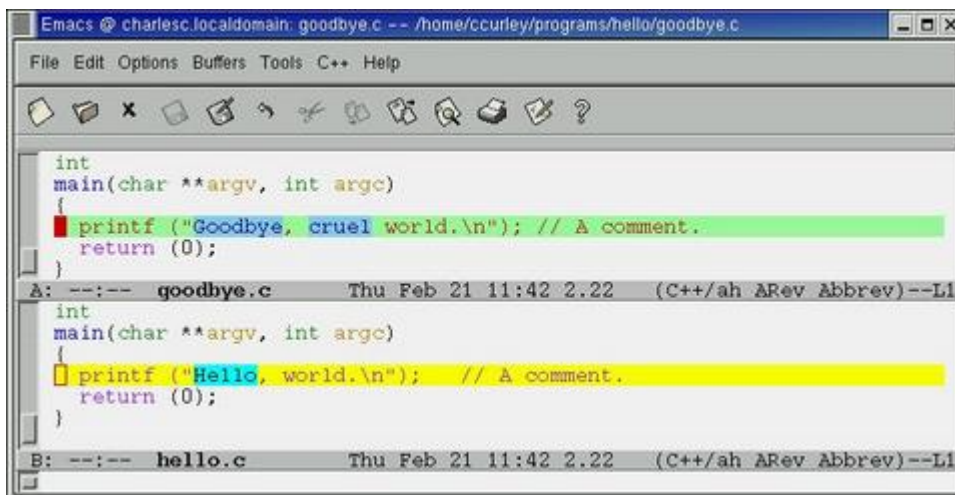


Figure 10. Showing the Difference between Two Files

Not only that, but you can use single-character commands to move differences from one file to the other; `a` or `b` moves the line from the A or B buffer, respectively, to the other. This allows you to walk through two files, compare them and accept or reject each difference.

To step through the two files one difference at a time, put the ediff control panel in focus. Use the spacebar to advance and the P key to go back (Figure 11). Ediff mode can keep up with changes you make on the fly. Try adding a line to goodbye.c right after the printf line, a call to flush();.

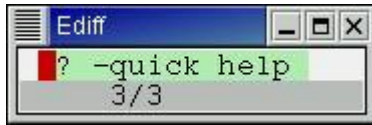


Figure 11. The Control Panel for Emacs' Ediff Mode

Help

Emacs has a huge library of built-in help. The gateway to Emacs' help is Ctrl-H. If you aren't familiar with Emacs at all, get started with the tutorial, Ctrl-H T. If you are already familiar with Emacs, you should walk through the tutorial; there's always something to learn about Emacs.

If you are familiar with the GNU program Info, you already know how to use Emacs' help system. Ctrl-H I gets you to the top menu of the Info system. From there, **Memacs** gets you to the Emacs documentation. And, of course, the documentation for Info is available from the top-level Info menu.

If you want documentation for other programs or for Linux function calls, you can use Emacs as an Info reader. Or, you can read the man pages from within Emacs. You can use Emacs as a front end for Man with the command M-X **manual-entry**. Or you can have Emacs interpret the man page and display it for you with the woman (WithOut MAN) package. Since the Cygwin tools for Windows (www.cygwin.com) include man pages but not a program to read them, woman is just the ticket.

For example, to see the man page for printf, put the cursor over the word printf in your source code. Enter M-X **woman** and press Return. Emacs will propose printf as the default manual entry. There are two printf man pages, one in man 3 and one in man 1. Tell Emacs you want man 3 by pressing 3 Tab. Emacs will complete the filename, then press Return. Emacs will show you the man page for printf.

Conclusion

Emacs is a fully integrated development environment. While a front end for external programs, Emacs uses free software for the back end. In that sense, it is better integrated into Linux than some proprietary IDEs. There is very little missing from Emacs, but if something is missing, Emacs is open-source and free software. You can write it.

[The Mode Line](#)

[Portability](#)

[Tab Completion](#)

[Remote Work](#)

[Resources](#)

email: ccurley@trib.com

Charles Curley (w3.trib.com/~ccurley) lives in Wyoming. He has 23 years' experience with computers, much of that as a software developer. He has worked for Hughes Aircraft, Hewlett-Packard, Microsoft and the Jet Propulsion Lab. He contributed to Sams' Teach Yourself Emacs in 24 Hours (ISBN: 0-672-31594-7).

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

[Advanced search](#)

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

Mediator/Python

Doug Farrell

Issue #98, June 2002

Using Python and the Mediator pattern to make dialog boxes that are smart and responsive without creating a coding mess.

I'll confess right now and tell you that I'm a newbie to Linux, having moved into a UNIX environment when I changed jobs almost two years ago. Before that I was an embedded systems programmer who worked primarily in a Windows environment. If you can find it in your heart to forgive me, I'd like to say the UNIX/Linux philosophy has been steadily winning me over. In addition to learning this amazing OS, I've had the opportunity to be exposed to the Python language. Not only do I find Python to be a powerful and expressive tool, but with the addition of wxPython as a GUI extension, I won't be going back to programming the Win32 API.

One of the things I did regularly in my GUI work was develop dialog boxes. Part of my design goal was to make the controls on the dialog boxes interactive, so the behavior of the controls would help my users get their work done. You've probably seen applications that do this, where one or more controls will change state depending on the state of some other control. This kind of interaction can be created by placing code in the event handlers of these controls. However, you'll end up writing a great deal of code in these event handlers to do all of this well. Plus, coding the interaction into the event handler couples the controls together. This means they have to know about each other in order to interact. If your dialog box gets at all complicated, it can lead to a maintenance nightmare. Just picture all of those controls making decisions about what their current state means to all the other controls on the dialog box. Talk about spaghetti code!

When I was creating this kind of dialog box I could see it would lead to a mess, so I wanted to decouple the controls from one another. I also wanted to centralize the interaction code in one place, so changes would have to be updated in only one place. The trick was to decide how to do this. One thing I

tried, and I'm sure many others do this as well, was to search the Web to see if anyone had done something similar. What I found were discussions of design patterns and, specifically, the book *Design Patterns: Elements of Reusable Object-Oriented Software*, which I highly recommend to anyone who takes their code seriously. The examples in the book are based on C++, but the concepts apply to many OO-based languages, including Python. The pattern we're going to use as a solution to our dialog box problem is called the Mediator pattern. This pattern encapsulates the interactions of a set of objects.

From an OO point of view, there is one Mediator object containing all the objects that we want to have interact with one another; they are called Colleagues. The Colleagues have a weak reference to the Mediator object and none to each other. The Mediator has strong reference to the Colleague objects and can update them directly in order to create the desired behavior for all Colleague objects. The benefits of this pattern are what we're looking for; it centralizes the interaction of objects and reduces the coupling between them.

The Mediator/Colleague pattern is implemented as an interface that can build actual class objects. The Mediator interface has a method called `ColleagueChanged()`, which is what all Colleagues call to inform the Mediator that a change has occurred. The Colleague interface has only one required method, called `Changed()`, which each derived object calls to inform the Mediator that a change in state has occurred. In addition, the Colleague base class has a public data member called `mediator`, which is a reference to the Mediator object that contains it.

All of that's very nice, but how do we actually implement a dialog box that uses the Mediator/Colleague pattern? We'll do this using Python's OO features and using wxPython as the GUI interface for a window. First things first, though; let's create a Mediator base class:

```
class Mediator:
    def __init__(self):
        pass
    def ColleagueChanged(self, control, event):
        self._ColleagueChanged(control, event)
    def _ColleagueChanged(self, control, event):
        pass
```

In this example the Mediator class is implemented as a Template pattern. This pattern allows us to separate the interface and implementation of the class. Users of this class call the `CreateColleagues()` method but override the `_CreateColleagues()` method in their derived classes. I won't discuss the Template pattern further, but it's another very useful pattern to know.

Now let's create our Colleague base class:

```

class Colleague:
    def __init__(self, mediator):
        self.mediator = mediator
    def Changed(self, colleague, event):
        self._Changed(colleague, event)
    def _Changed(self, colleague, event):
        self.mediator.ColleagueChanged(colleague, event)

```

The Colleague base class is also implemented as a Template pattern. As before, users call the Changed() method but override the _Changed() method, if necessary. In addition, the Colleague has a data member, self.mediator, which is a reference to the containing Mediator instance. This reference is passed into the constructor of the Colleague.

Since our example program is intended to show the utility of the Mediator/Colleague pattern, it's somewhat contrived. To make the example a little simpler I've based the one window of the example on wxFrame from the wxPython library, rather than wxDialog. Otherwise, the code is the same. Because our example program uses wxFrame as the container of the controls, it's the logical choice to be the Mediator. In order to give wxFrame the interface of the Mediator class, we'll create a new class MainFrame that looks like this:

```

class MainFrame(wxFrame, Mediator):
    def __init__(self, parent, ID, title):
        wxFrame.__init__(self, parent, ID, title,
            wxDefaultPosition, wxSize(400, 300))
        Mediator.__init__(self)

```

In this code we've created a new class, MainFrame, that inherits from both wxFrame and Mediator base classes. For this to work in Python we have to call explicitly the constructor of both the parent classes. This call is made in the __init__() method of the MainFrame class.

In order for our MainFrame class to interact with its controls as Colleagues, those controls have to be derived from the Colleague base class. As an example, let's create a text control that is also a Colleague object. We do this by creating a new class, myTextCtrl, that looks like this:

```

class myTextCtrl(wxTextCtrl, Colleague):
    def __init__(self, mediator, *_args, **_kwargs):
        apply(wxTextCtrl.__init__, (self,) + _args,
            _kwargs)
        Colleague.__init__(self, mediator)

```

Here we've created a new class that inherits from both wxTextCtrl and Colleague base classes. Again, in order for this to work in Python we have to call explicitly the constructor of both the parent classes. This is done in the __init__() method of myTextCtrl. In order to get all the optional parameters of the wxTextCtrl class properly passed to its constructor, I'll use the apply() function to call its __init__() method and pass in the parameters. The __init__() method of the Colleague base class is called directly, passing mediator as a parameter.

Now we have a class that is both a Colleague and a wxTextCtrl. An instance of this class will receive all the events generated by a wxTextCtrl and also has the behavior of the Colleague class. When an event occurs that our MainFrame object cares about, the event handler calls the Changed() method and passes the self reference and the event as parameters. As defined in the Colleague class, the Changed() method calls the ColleagueChanged() method of the control's Mediator reference. In this way the MainFrame object (which is a Mediator object) is informed of all changes occurring in its contained controls.

So how do we tie all this together in our MainFrame window? First, as we did with myTextCtrl, we must create derived classes of all the controls that will interact on our window that also derive from the Colleague base class. Then, as in most wxPython windows, we must create our controls in the window's constructor; in this case MainFrame's __init__() method. Each time a control is created, MainFrame passes itself as a parameter to the derived control. You'll see this in the complete example program that accompanies this article [available at <ftp://linuxjournal.com/pub/lj/listings/issue98/5858.tgz>]. Don't be dismayed by the amount of code in the MainFrame.__init__() method; a great deal of it calls layout functionality provided by wxPython and is not strictly necessary for our example. It just makes it look nicer.

One thing you should notice I've done in the MainFrame.__init__() method is create a dictionary object called self.__colleagueMap. I've placed sets of key/value pairs into this dictionary consisting of a reference to the created Colleague controls and a method of the MainFrame class. I've done this because Python does not have a switch/case construct like C/C++ has. This dictionary provides an elegant mechanism to call the correct method whenever a Colleague object has changed, without resorting to a lengthy if/elseif construct. You'll see this in the example program in the implementation of the _ColleagueChanged() method, as shown below:

```
def _ColleagueChanged(self, colleague, event):
    if self.__inProcess != True:
        self.__inProcess = True
        if self.__colleagueMap.has_key(colleague):
            self.__colleagueMap[colleague](event)
        self.__inProcess = False
```

In this code the parameter, colleague, is used as the lookup into the dictionary. If the colleague exists as a key, then the corresponding method is called and the event is passed as a parameter. This is a very slick way to implement a multiway branch like the switch/case construct.

Now our Mediator and Colleague objects exist and are connected together. The Mediator object (MainFrame) will be notified of any relevant event generated by a Colleague object (control). So what's left to do? We have to provide the centralized code that will create the desired interaction between our controls.

This work is done in the methods we've placed into the self.__colleagueMap dictionary. Into each method we place the code that reacts to that controls event. Since these methods are part of our Mediator (MainFrame) object, they know about and have access to all other controls on the window.

The example program has been tested under Python versions 2.1 and 2.2, along with their respective wxPython versions. When the example program is run you should see a window open up that looks like the one pictured in Figure 1.

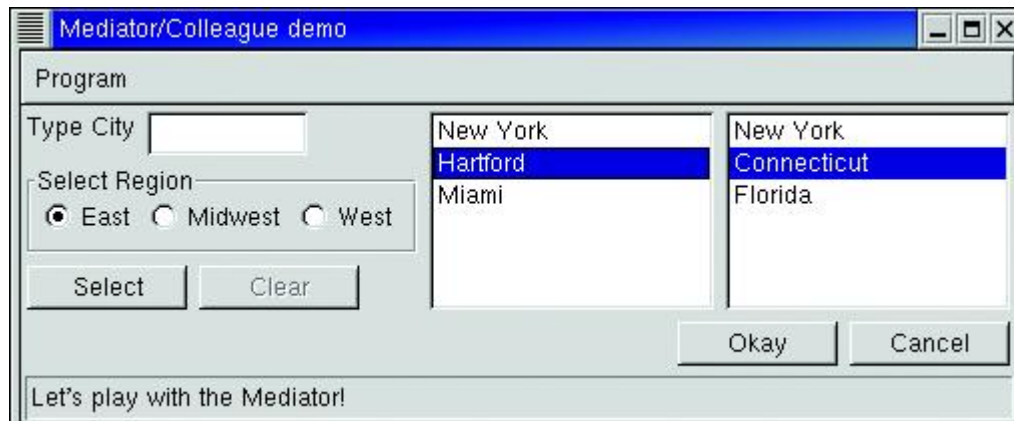


Figure 1. Mediator Screenshot

The interaction on the displayed window involves most of the controls. When you type a character in the text box, the program does some simple speed selection and highlights the first entry in the list box that matches. In addition, the Select and Clear buttons become enabled. If you select either a city or state, the complementary selection is made in the other list box. If you select a different region with the radio buttons, that action re-initializes the lists boxes, clears the text box and disables the Select and Clear buttons. Clicking on the Clear button alone clears the text box and disables itself. All of the interaction code is in the MainFrame class, and the controls are not coupled to one another.

Our wxFrame window doesn't do anything very useful, but it does demonstrate how the Mediator pattern can orchestrate the behavior of controls on a window. The real payoff for this effort comes when you apply this pattern to a more involved dialog box, where the interaction complexity can grow at a shocking rate. To manage it, all we have to do is add another control that is derived from the Colleague class and add the corresponding interaction code to our ColleagueChanged() method, and the complexity is handled.



email: writeson@earthlink.net

Doug Farrell is a Senior Software Engineer with Scholastic, Inc. in their Connecticut office. He develops web applications to put reference titles on the Internet. When not scratching his head trying to solve some programming puzzle, you can find he and his wife, Susan, out on their bicycles cranking out the miles.

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

[Advanced search](#)

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

Advanced search

Python 2.2 Q&A with Guido van Rossum, Creator of Python

Wesley J. Chun

Issue #98, June 2002

What's new, what's planned and why people need to calm down about the division operator.

With version 2.2 [and now 2.2.1] finally released to the world, we took some time to ask Guido van Rossum, Python's inventor and PythonLab's director, to share some of his thoughts about the latest Python news.



Wesley Guido, thanks for taking the time today to talk with us at *Linux Journal*.

Guido It's always a pleasure to work with *LJ*.

Wesley Give us one or two sentences of overview about the new 2.2 releases.

Guido The release of 2.2 is the first step on the way to a new object model for Python, where built-in classes and user-defined classes finally have equal footing. In addition, there's cool stuff, particularly generators.

Wesley Why should current developers migrate from Python 2.0.x or 2.1.x to 2.2.x?

Guido To benefit from the new style class features, such as properties, static methods and subclassing list and dict, or for generators.

Wesley Why should current developers migrate from Python 1.x to 2.2.x?

Guido In addition to the features already mentioned, to benefit from the tons of bugs that were fixed and the features added in 2.0, such as list comprehensions, Unicode, XML and a new regular expression implementation. If you're happy using 1.5.2, be my guest. If there's anything in 1.5.2 that irks you, there's a good chance it's been fixed in 2.x.

Wesley Some developers have never seen iterators or generators before. Where did they come from, and why were they added to Python?

Guido C++ has iterators, and they are actually a pretty common programming pattern, just not usually a language feature. But in an interpreted language like Python, the boundary between the language and the library is often blurred. Generators come from a little-known language called Icon, but they deserve to be in the spotlight; they are an incredibly powerful way to write more readable code for certain situations. You can also see them as a tamed form of co-routines, an old idea (described, for example, by Knuth) that unfortunately went out of style because it required stack fiddling.

Wesley In many other object-oriented languages, classes are seen as types and their instances as objects of those types. Why was this not the case for Python and why the turnaround?

Guido That was an implementation artifact. The very first version of Python (which was never released, but used internally at CWI in early 1990) didn't have a class statement; the only way to add a new data type to a program was to write an extension in C. Someone suggested that I add classes so that end users also could define new data types. I invented an implementation that required minimal changes to the abstract Python virtual machine. But the class statement couldn't do exactly the same things as the C programmer could. I've known for a long time that this needed to be fixed, but solving it required a large implementation effort for which I didn't have the time earlier on—in part because during the intervening ten years the language had become so successful that I just couldn't change the rules. I had to make sure that old code continued to work unchanged, and I had to provide a migration path to using the new features. I think I've succeeded at both.

Wesley The changing of the division operator has been the source of much controversy and debate. What are you telling each side to keep flame wars to a minimum?

Guido The folks who want the new division operator don't need to be told to keep flame wars at a minimal level—it's the people who mistakenly fear that all their code will break overnight who need to be calmed down. It's the same issue as with the new style classes: 99% of the implementation effort went into finding ways to implement it without breaking old code. No code using `/` for int division will have to be changed until Python 3.0, which is several years off. In the meantime, folks can choose on a per-module basis to use `/` for float division. I'm also supplying two tools (`finddiv.py` and `fixdiv.py`) that can be used to track down and fix `/` operators that would need to be changed to `//` in order to work properly in Python 3.0. We've already converted the entire standard library to using `//` where necessary. It's not a particularly hard task, but it can't be 100% automated because it's impossible to tell for sure at compile time what the operand types are. By the way, a change to integer semantics that is nearly universally liked is the automatic conversion to the long type when any operation on ints would raise an `OverflowError`.

Wesley Any new developments for those who write Python extensions?

Guido Yes, they can create types that are subclasses of standard types, and they can create types that are subclassable. There's a bunch of new slots in the type object that extension authors can use to fine-tune what happens when an object of their type is created or destroyed.

Wesley Python 2.2 brings a good number of new and improved modules. What are your top ten updates to the Python Standard Library?

Guido Surprisingly, I'm not usually on the bleeding edge of the library, so I can't quite make it to ten. First of all, there's the new e-mail package by Mailman author Barry Warsaw. There's the hot-shot profiler by my colleague Fred Drake, the XML-RPC support by Fredrik Lundh and, last but not least, the IPv6 support, a coproduction of Jun-ichiro "itojun" Hagino and Martin von Loewis. I think Python's IPv6 support is way ahead of other languages.

Wesley Are the new features of 2.2 going to be included in Jython, the Java language Python interpreter?

Guido Yes, the Jython developers have been active on the python-dev mailing list to be sure that we wouldn't add features that were unimplementable in 100% pure Java. This is one of the reasons why generator functions require the use of a new keyword—Jython needs to know that something is a generator at compile time. They released their final Jython 2.1 soon after we released Python 2.1, so I expect that they are working on Jython 2.2. The type-class unification work should be easier in Jython than it was in the C implementation of Python,

because they were already basing both built-in types and user-defined classes on Java classes.

Wesley Jython 2.1 was released soon after Python 2.2. What version of Python is it most compatible with? In other words, how far behind is Jython?

Guido The Jython version numbers match the Python version numbers, so Jython 2.1 is compatible with Python 2.1.

Wesley What can we look forward to in Python 2.3? Any timeline for that release?

Guido I expect 2.3 to be a release of consolidation and performance tweaks, not of grand new features like 2.2 was. We may also focus more on the standard library than on changes to the core language. There's now a schedule for 2.3; see PEP 283 for tentative future release dates. I like doing a new release every six months, as we've done since 2.0. We are already done with 2.2.1, which is a pure bug-fix release for 2.2, as 2.1.2 is a pure bug-fix release (the last one, I'm sure) for 2.1. This is important because a lot of third-party code is out there (Zope 2.5 for example) that depends on 2.1 and cannot be migrated to 2.2 right away.

Editor's note: look for Wesley's article in next month's *LJ* for further details on the advancements of Python 2.2.



Wesley J. Chun, author of *Core Python Programming*, has over a decade of programming and instructional experience. He is employed at Synarc, a service company that utilizes Python to develop applications that allow radiologists to perform patient assessments. He can be reached at cyberweb@rocketmail.com.

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

[Advanced search](#)

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

[Advanced search](#)

The OSCAR Revolution

Richard Ferri

Issue #98, June 2002

Richard describes the history and goals of the Open Source Cluster Application Resource.

“Serve no whine before its time” is a bad pun attributed to Rob Pennington of NCSA at the very first OSCAR meeting, held in April 2000 at a hotel a stone's throw from Oak Ridge National Lab. A varied cast representing the national labs, academia and industry was assembled to discuss what was known at the time as the CCDK (Community Cluster Development Kit), which would morph into the OCG (Open Cluster Group) and their first project, OSCAR (the Open Source Cluster Application Resource).

The cast had broken clusters down into components and had assigned “czars” (leaders) and “whiners” (interested parties) for each component. The czars were to lead each component group, and the whiners were to whine loudly and often enough to make sure things got done on schedule, meeting the group's requirements. From that very first meeting when the czars and whiners were named, it was clear that OSCAR development would be different from all other software development that had gone before. After all, where else would one find companies like IBM, Dell, SGI and Intel working closely together to produce open solutions in a hotly contested space like clustering?

The original idea for OSCAR came about over dinner at a DOE-sponsored cluster meeting at Argonne National Lab, where Dr. Timothy Mattson, a research scientist at Intel, and Dr. Stephen Scott, a research scientist at Oak Ridge National Lab, discussed the problem of getting Linux clusters accepted into the mainstream. The problem, they decided, was that it was just too difficult for noncomputer programmers to assemble their own cluster. Books like *How to Build a Beowulf* (Sterling, et. al.) would help the computer savvy understand the concepts and construct his or her first cluster, but there were still daunting problems. There was an enormous amount of code to download, all at differing levels of reliability, support, integration and documentation.

Sometimes the documentation for various packages was dated and contradictory. There were many Linux distributions to choose from, each trying to distinguish themselves by being slightly different from the next distribution. This meant that some commands worked differently or that different packages had to be installed to get a service to work properly.

The problem, they decided, was that with everyone trying to build their own cluster to tap into cheap cluster computing, each cluster was being built from scratch. There had to be some economy in compiling the best available software, practices and documentation in a single spot, integrating the package on different types of hardware and making it available to users for free (as in free beer). This concept, making clusters easy to build for the nonprogrammer, is a central tenet of OSCAR.

First Meeting

The historic first meeting in Oak Ridge was attended by Tim Mattson and Stephen Scott, the leaders of the OCG; Gabriel Bonner from SGI; Dave Lombard from MSC.Software; Rob Pennington of NCSA; Greg Lindahl, now of Conservative Computers; Ken Briskey and myself from IBM; Greg Astfalk from HP; and Clay Taylor from MPI Software Technologies. Shortly after the first meeting, Broahn Mann from Veridian joined to bring his parallel scheduling skill to the team, as did Jeremy Enos and Neil Gorsuch from NCSA (who implemented SSH on OSCAR) and Mike Brim from Oak Ridge National Lab (who wrote most of the integration scripts and packaging). Most recently, Jeff Squyres and Brian Barrett from Indiana University joined the OSCAR Project representing LAM/MPI. The disparate group agreed on three major core principles:

1. That the adoption of clusters for mainstream, high-performance computing is inhibited by a lack of well-accepted software stacks that are robust and easy to use by the general user.
2. That the group embraces the open-source model of software distribution. Anything contributed to the group must be freely distributable, preferably as source code under the Berkeley open-source license.
3. That the group can accomplish its goals by propagating best-known practices built up through many years of hard work by cluster computing pioneers.

With these principles firmly in place, the group used a divide-and-conquer method to list the components that comprise clusters. The component groups decided on the best-known, open-source solutions for each component and presented the information to the group at large. Taken collectively, these best-known practices for each component comprised a viable cluster solution. Even

with the component solutions in hand, there was a massive and time-consuming integration effort by Oak Ridge National Lab, led by Mike Brim and Brian Luethke, and a separate test effort, which was led by Jenwei Hsieh, Tau Leng and Yung-Chin Fang from Dell. Through their efforts, and face-to-face and remote-integration parties, OSCAR eventually morphed into something to share with the rest of the community.

The Software Stack

It took nearly a full year, but OSCAR had a beta demonstration at SC2000 in Dallas, Texas at the Oak Ridge National Lab booth in November 2000. The beta was run on a heterogeneous cluster of servers provided by Dell and SGI. The first release was announced shortly thereafter and made a successful debut at LinuxWorld Expo in New York City in February 2001, at the Intel booth. Since then, there have been continuous improvements in the OSCAR software stack, which currently includes:

- **Linux installation:** SIS (system installation suite). SIS is an open-source cluster installation tool based on the merger of LUI (the Linux utility for cluster install) and the popular SystemImager. SIS, developed by Michael Chase-Salerno and Sean Dague from IBM, made its debut in the 1.2.1 version of OSCAR. Most recently, Brian Finley of Bald Guy Software, the creator of SystemImager, has been attending the OSCAR meetings and looking for free beer, as in free beer.
- **Security:** OpenSSH—the most common way to allow secure connections in a Linux environment. OpenSSH is a collection of packages that handles secure connections, server-side SSH services, secure key generation and any other functions used to support secure connections between computers.
- **Cluster management:** for cluster-wide management operations, OSCAR uses the Cluster Command and Control (C3) management package developed at Oak Ridge National Lab by Stephen Scott and Brian Luethke, an East Tennessee State University student working at ORNL. C3 provides a “single-system illusion” so that a single command affects the entire cluster. C3 remains installed on the cluster nodes for later use by cluster users and administrators.
- **Programming environments:** Message-Passing Interface (MPI) and Parallel Virtual Machine (PVM). Most cluster users write the software that runs on the cluster. There are many different ways to write software for clusters. The most common approach is to use a message-passing library. Currently, compilers or math libraries installed by OSCAR come from the Linux distribution. Both LAM/MPI and MPICH have been available since OSCAR 1.1.

- Workload management: Portable Batch System (PBS) from Veridian and Maui Scheduler (developed by Maui High Times Computing Center). To time-share a cluster, some type of workload or job management is needed. Maui acts as a job scheduler for OSCAR, making all resource allocation and scheduling decisions. PBS is the job server/launcher and in addition to launching and killing jobs, handles job queues.

MSC. Software and OSCAR

MSC.Linux, a distribution developed by the Systems Division of MSC.Software Corporation, is of special importance in the acceptance of OSCAR. Shortly after the 1.0 version of OSCAR was available, MSC.Software announced their own cluster solution, the MSC.Linux Version 2001 operating system. This 2001 offering was in large part based on OSCAR, the first commercial offering based on the work of the OCG. MSC.Software's Joe Griffin added a Webmin interface to LUI (the first OSCAR cluster installation tool), which generated LUI bottom-line commands for multiple nodes to provide an easy-to-use interface in defining the nodes of the cluster and what resources to install on each. One of the original intents of the OCG was that commercial companies would see the value in the open OSCAR software stack and build their own proprietary or open stacks around the OSCAR stack. In so doing, companies using OSCAR would be freed from the mundane chores associated with building a cluster, such as providing the basic infrastructure, and could concentrate instead on more cutting-edge improvements to distinguish their offering.

Working Together

Like other far-flung open-source projects, it was clear from the beginning that doing the work of the consortium face to face would not always be an option. The travel expense was simply too great, and it was difficult to align so many schedules. To coordinate the work, the group held open weekly phone conferences and would rely on mailing lists and an occasional meeting at a workshop or expo. There were face-to-face "integration parties" held quarterly, one at Intel in Hillsboro, Oregon and another at NCSA in Illinois. But for integrations held between meetings, a new construct was developed, called DIP Day, for distributed integration party. The intent of DIP Days was that everyone working on the project that had a cluster would set aside those days to work on OSCAR, jointly and remotely. Everyone would download the OSCAR package and install and run it, reporting any bugs to the group. On DIP Days, programmers were expected to provide fixes in real time, so that multiple iterations of the code could be tested shortly. Several conference calls with the entire team were held every DIP Day to assess progress and assign new work and priorities. By loosely coordinating the group between DIPs and face-to-face meetings, OSCAR made great strides in reliability and function.

The OSCAR Experience: First Impressions

The first thing one notices when untarring the OSCAR file is that the OSCAR integration and test team has done a thorough job; there is extensive documentation on how to install OSCAR, the system requirements, the licensing (GPL) and the theory behind OSCAR itself. There is a quick start guide for the impatient cluster administrator, as well as a full descriptive text. One also notices that there's nothing additional to download; it's all included in the single OSCAR tar file. OSCAR takes the traditional view of clusters—a single server with N compute nodes; the server is responsible for installing, scheduling and monitoring the compute nodes. Nodes in the cluster should be running homogeneous software, meaning the same distribution and version of Linux. The first command the user enters is **install_cluster**, which does a multitude of things: creates necessary directories; manages NFS and xinetd; installs LAM/MPI, C3, PBS, Maui, OpenSSH, SIS, Perl, SystemImager and MPICH; updates various profiles and configuration scripts; and launches the OSCAR wizard.

If all goes well, you're in for a pleasant surprise, namely, the OSCAR wizard. The OSCAR team felt the wizard would be another distinguishing feature of OSCAR in the field of Linux cluster solutions. The purpose of the wizard is clear—follow the wizard and you too can install a cluster painlessly. Each step along the wizard's path has entry and exit criteria. Once the exit criteria is successfully met, OSCAR gives a success message to indicate it's safe to move on to the next step.

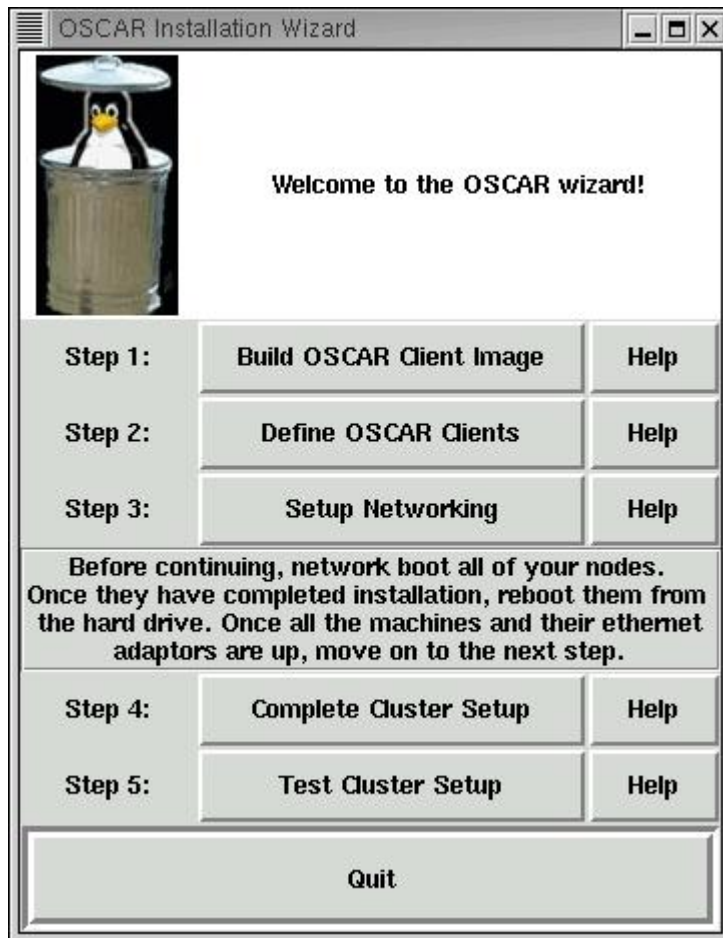


Figure 1. The OSCAR Wizard

Following the wizard, pressing the Build OSCAR client image button brings up the second panel, the Create a SystemImager Image panel.

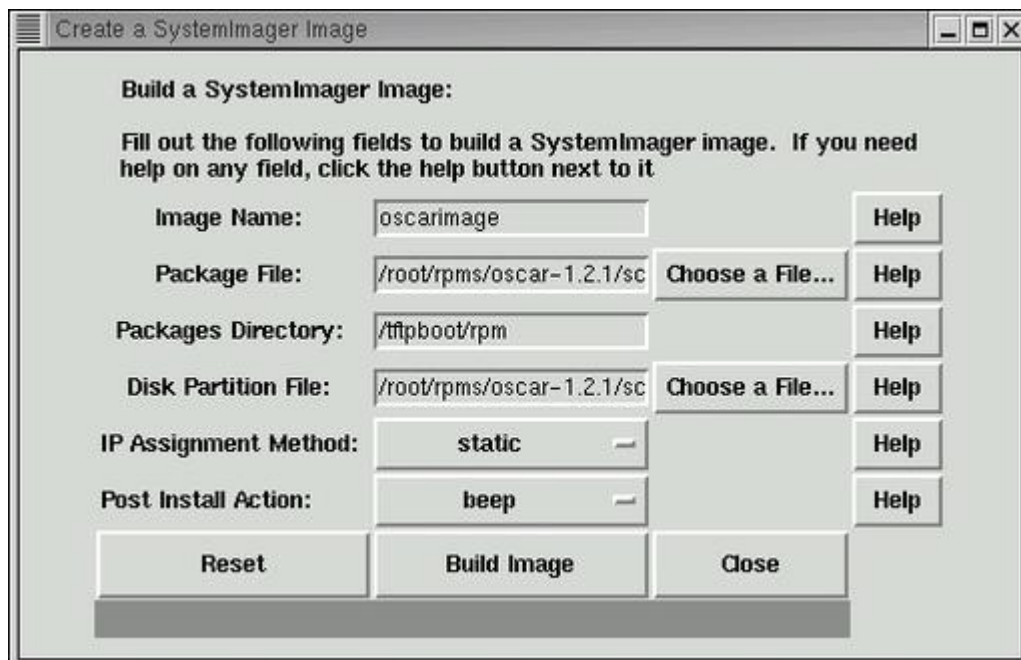


Figure 2. Building a SystemImager Image

The purpose of the SystemImager panel is to create a filesystem on the server that will later be installed on each client. The Image Name field allows the user to create multiple SystemImager images, each with a unique name. The Package File field provides a list of packages that will be installed on the client; OSCAR provides sample lists that meet most user requirements. The Packages Directory tells where the RPMs are coming from, and the Disk Partition File field allows the user to customize the disk partitions. Again, OSCAR provides default disk partition definition files for both IDE and SCSI drives. Pressing the Build Image button starts the process of building a client image on the server. Once complete, it's time to go back to the wizard for step two, defining the OSCAR clients.

Field	Value	Action
Image Name:	oscarimage	Help
Domain Name:	pok.ibm.com	Help
Base Name:	oscarnode	Help
Number of Hosts:	0	Help
Starting Number:	1	Help
Starting IP:	192.168.64.209	Help
Subnet Mask:	255.255.255.0	Help
Default Gateway:	192.168.64.208	Help
Reset		Addclients
		Close

Figure 3. Adding Clients to a SystemImager Image

From the Add Clients panel, the user can specify a range of IP addresses to be associated with a list of new clients. Each client is associated with an image name using the Image Name field. One can define a set of clients in a range of IP addresses, each having the same netmask and default gateway. Pressing the Addclients button builds client definitions for SIS. Once complete, it's back to step three on the wizard, Setup Networking.

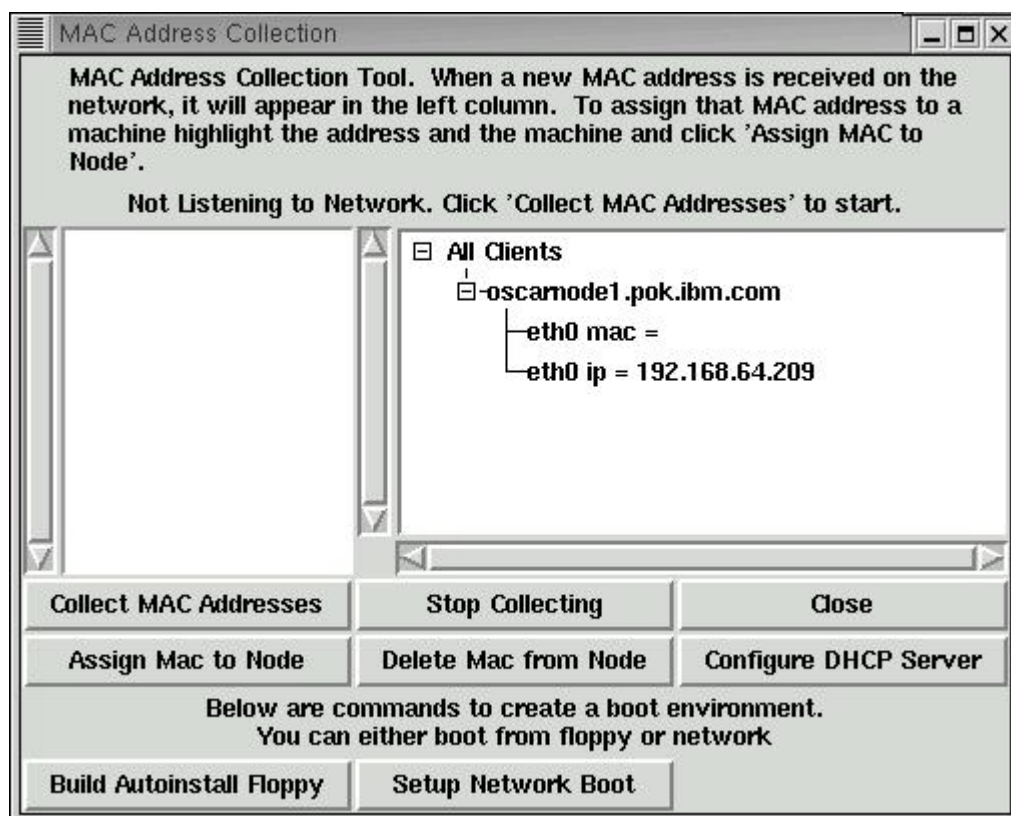


Figure 4. MAC Address Collection

From the Setup Networking panel, MAC addresses are collected for each client in the cluster. If the node is capable of true network (PXE) boot, you simply associate a MAC address with a client, and you're ready to power up the node. If the node is not PXE-enabled, you can write a SystemImager boot diskette from the Build Autoinstall Floppy button. Once the MAC addresses are collected, it's time to press the button to Configure the DHCP Server and boot all the nodes to initiate Linux installation.

Once all the nodes are installed, each node starts this really annoying and incessant beeping telling the system administrator to pop out the diskette or turn off PXE and reboot the node from the hard drive. Once they are all booted, the nodes are ready to Complete Cluster Setup from the wizard (really just syncing the time between servers and clients and running any package-sensitive postinstallation scripts). The Test Cluster Setup button from the wizard runs short jobs, checking each flavor of scheduler and parallel library.

Once the cluster is fully installed and functioning, there are test scripts to check the overall health of the cluster. Running the test_install script will check to make sure PBS or Maui Scheduler is configured and running, that the C3 tools are installed and that the cluster at that time is ready to start accepting parallel jobs.

OSCAR Futures

As of this writing, OSCAR 1.2.1 is available, which runs under Red Hat Linux 7.1. MSC.Linux Version 2001 is based on OSCAR 1.1. The 1.x1 releases were quite popular in the community—they've been downloaded roughly 25,000 times from SourceForge. However, the biggest problem OSCAR is now facing is that it takes the relatively few OSCAR developers time and effort to integrate all the new software packages that want to be included in OSCAR. The OSCAR 2.0 effort is underway, with an emphasis on establishing component APIs so that anyone with an open software package can integrate their package with OSCAR. The OCG itself is growing. Since Tim Mattson went on Intel sabbatical, Jeff Squyres from the University of Indiana has taken the leadership role in the OSCAR 2.0 architecture and consistency. Ibrahim Haddad from Ericsson [and a frequent *LJ* contributor] has joined the consortium with interesting ideas on how to bring OSCAR to near-telecom levels of reliability. Jim Garlick, representing Lawrence Livermore National Lab also has joined the consortium bringing his real-world large cluster scaling experience and concerns to the group.

Platforms and Distros

At the very first OSCAR meeting, it was agreed that OSCAR should not be tied to a particular Linux distribution or platform. However, to date the OCG's efforts have been largely focused on the Red Hat and MSC.Linux distributions and the IA-32 architecture. This focus will expand in 2002. The purpose of integrating SIS into OSCAR was to be able to support all RPM-based distributions: SuSE, Turbolinux, Red Hat, MSC.Linux and Caldera, and later to support deb-based distributions such as Debian. Additionally, the architecture of SIS makes it easy to port to new platforms. NCSA already has a beta version of OSCAR running on Itanium, and Oak Ridge has done extensive testing with Red Hat 7.2. Given the open API and ability to run on many different distributions and platforms, expect OSCAR and the OCG to expand dramatically this year.

Final Thoughts

The impacts of OSCAR on the Linux clustering community can be viewed in several different perspectives. At the most apparent, OSCAR is providing a useful clustering tool that is usable across various manufacturers' platforms. It has removed much of the ambiguity inherent in assembling software from various web sites by putting together a single, integrated, documented, tested and supported package. It is truly a clustering solution that a nonprogrammer can implement. However, beneath the surface, the OCG is a thriving consortium composed of national labs, academia and industry that are cooperating to bring new open-source solutions to Linux. Along the way, the consortium had to break new ground in ways to cooperate, with unique

concepts like DIP Days. In retrospect, the consortium's most important long-term contribution to the community may be in developing new ways to work together for the betterment of open source.

Resources



email: rcferri@us.ibm.com

Richard Ferri is a senior programmer in IBM's Linux Technology Center, where he works on open-source Linux clustering projects such as LUI and OSCAR. He now lives in a rural setting in upstate New York with his wife Pat, three teen-aged sons and three dogs of suspect lineage.

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

[Advanced search](#)

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

[Advanced search](#)

A NATural Progression

David A. Bandel

Issue #98, June 2002

David continues his series on the Netfilter framework with a look at NAT and how to avoid common errors when constructing iptables rules.

Editor's Note: Due to a printer error, David Bandel's article on iptables building was not complete in the magazine. We present it here in its entirety.

One of the best tools at our disposal within the Netfilter framework is NAT. NAT allows us to obfuscate (but not hide) our true network, forcing would-be black hats to work harder (and possibly go after easier targets). It also permits us to make the best use of limited IPs. Last month [see "Netfilter 2: in the POM of Your Hands" in the May 2002 issue of *LJ*] we looked at extending iptables to include experimental or beta matches and targets. This month we look at doing NAT the correct way, take a closer look at one or two other matches, then see what some of the more common errors are when constructing iptables rules and how to avoid them.

SNAT, DNAT and Managing Services

Given the shortage of usable IPv4 addresses left today, it's likely your ISP didn't provide you enough IPs to run all your systems. If you're lucky, you got more than half a dozen you could actually use. But if you don't use them, you'll lose them. And you don't want that, or there's no room for expansion tomorrow. So you're going to make your firewall look like as many systems as you have IPs to use. How to do that? The easiest way is to assign all your IPs to one NIC (the one connected to your ISP) and SNAT connections so they look like they come from each IP in turn (this example assumes your internal network is 192.168.0.0/24, which is bound to eth1, and your usable IPs are 209.127.112.26-209.127.112.30, which are bound to eth0):

```
iptables -t nat -A POSTROUTING
-o eth0 -s
--to-source 209.127.112.26-209.127.112.30
```

Now iptables will NAT the first connection to .26, the second to .27, the third to .28 and so on, wrapping around to .26 after the connection to .30.

One word of caution: test this before you deploy it. I've had one router that didn't like seeing multiple IPs sourced from the same MAC address. It would pass the first connection, but subsequent connections would time out. The router's built-in firewall (which couldn't be turned off by the client) most likely thought the other packets were spoofed and was silently dropping them.

Let's make sure we accept all outgoing connections but only accept incoming connections that are related to these outgoing connections:

```
iptables -t filter -A FORWARD -i ! eth0 -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -m state
--state ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -m state
--state NEW,INVALID -j DROP
```

Now that we've managed outgoing traffic, let's assume we've moved all our services inside this firewall. We'll further assume they are all inside eth1 on the 192.168.0.0/24 network. Each service has two IPs associated with it: an external IP that the world sees and an internal IP that we see. Specifically, we'll assign the following:

- Apache web server (serves both insecure and secure connections): 209.127.112.26 and 192.168.0.4
- FTP server: 209.127.112.27 and 192.168.0.5
- Primary DNS server: 209.127.112.26 and 192.168.0.6
- Secondary DNS server: 209.127.112.28 and 192.168.0.7
- Primary mail server: 209.127.112.28 and 192.168.0.8
- Secondary mail server: 209.127.112.29 and 192.168.0.9

Because of our iptables state table rules above, each service (or more specifically, each port that corresponds to that service) will not only have to be forwarded through the firewall to the correct IP inside, but we'll need a rule to accept that NEW traffic. Starting with Apache, which in our case uses both ports 80 and 443 (for SSL), we have:

```
iptables -t filter -I FORWARD -i eth0
-d
iptables -t nat -A PREROUTING -d
-p tcp --dport 80 -j DNAT --to-destination
iptables -t filter -I FORWARD -i eth0
-d 209.127.112.26 -p tcp --dport 443 -j
ACCEPT
iptables -t nat -A PREROUTING -d
--dport 443 -j DNAT --to-destination 192.168.0.4
```

Notice that we had to insert a rule in the FORWARD chain. This is because we already have a more general rule that would have dropped NEW connections. We can insert a rule anywhere in a chain, but if we don't specify where, the default is to insert it as the first rule. Normally, this will not be a problem and will put our specific rules ahead of our general rules.

Note that we also specified the IP on which this connection should show. This is not necessary because connections showing up on other IPs will be handled by the state table and dropped, unless we've done something really questionable and started a web server on our firewall. If no ports on our firewall are open, we're okay. We always can protect them just in case by ensuring we also have stateful rules for our INPUT chain:

```
iptables -t filter -A INPUT -i ! eth0 -m state
--state NEW,RELATED,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -i eth0 -m state
--state RELATED,ESTABLISHED -j ACCEPT
iptables -t filter -A INPUT -i eth0 -m state
--state NEW,INVALID -j DROP
```

The first rule above allows NEW, ESTABLISHED and RELATED connections from lo (the localhost interface) as well as any internal devices, omitting only our external device, which is dealt with in the next two rules.

Next we look at the FTP connection. This is straightforward and exactly the same as the above rules, but for port 21:

```
iptables -t filter -I FORWARD -i eth0 -p tcp
--dport 21 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0
-d
--to-destination 192.168.0.5
```

We don't have to worry about the FTP-data channel (port 20) because our FTP server opens it outgoing, and our state rules will pass this new connection out.

Now it gets a little more difficult. DNS works on both UDP for normal queries and TCP for zone transfers. If we don't want to allow zone transfers to the outside, we only open UDP. If we want to allow zone transfers, then we have to allow both. Assuming we want to allow both, we know that we can specify UDP, TCP or ICMP as protocols. You must specify -p (protocol) in order to specify a port. If you want both UDP and TCP, you should be able to say "not ICMP", and the other two are assumed automatically. Unfortunately, it doesn't work that way.

When you specify a protocol, even if you say **-p ! ICMP**, it's the ICMP match module that is loaded, not the TCP and UDP match modules. So you'll get an error message when you specify a port. This is a danger with using a negative match; the match module that is loaded is the module specified, not the

modules you may assume are loaded. You must specify positively each match you want so the corresponding match module is loaded.

For now, let's assume you are only interested in opening the UDP port:

```
iptables -t filter -I FORWARD -i eth0
-d
iptables -t nat -A POSTROUTING -i eth0
-d
--to-destination
iptables -t filter -I FORWARD -i eth0
-d
iptables -t nat -A POSTROUTING -i eth0
-d
--to-destination 192.168.0.8
```

And finally, we need to deal with our mail host:

```
iptables -t filter -I FORWARD -i eth0
-d 209.127.112.28 -i eth0 -p TCP --dport 25 -J
ACCEPT
iptables -t nat -A POSTROUTING -i eth0
-d
--to-destination
iptables -t filter -I FORWARD -i eth0
-d 209.127.112.29 -i eth0 -p UDP --dport 25 -J
ACCEPT
iptables -t nat -A POSTROUTING -i eth0
-d
--to-destination 192.168.0.9
```

We can now accept incoming mail. Does anyone see a problem here?

If we test our outgoing mail using **mail -v user@another.dom**, our firewall will grab one of 209.127.112.25-209.127.112.30. If our DNS records say our MX host is 209.127.112.28, then we have only a 20% chance of grabbing that IP and an 80% chance that upstream mail hosts will bounce our mail as not being from a host with a DNS MX RR—not good.

So how do we fix this? If we have the luxury of adding *all* our IPs as MX hosts, that would solve part of the problem, but then upstream hosts might spend time connecting to IPs of ours that don't DNAT the mail through. And we really don't want all those IPs to appear as MX IPs.

The correct response is to add a more specific SNAT rule ahead of the general SNAT rules that will handle outgoing traffic on port 25. The danger here is that if we can't trust our internal users, we also must make sure that internal users can't abuse port 25. So we'll add three rules for outgoing traffic, one to SNAT port 25 traffic coming from our primary mail server (192.168.0.8) only out through 209.127.112.28, and two to block port 25 traffic from all other internal addresses except our true mail host:

```
iptables -t filter -I FORWARD -i eth1
-s
iptables -t filter -I FORWARD -i eth1 -p tcp
```

```
--dport 25 -s  
-I POSTROUTING -o eth0 -p tcp --dport 25  
-s 192.168.0.8 -j SNAT --to-source 209.127.112.28
```

Some of you may think: hah! caught him. The first two rules above are reversed. Well, yes, they are. But that's because we're inserting them one at a time as the first rule, so rule two above will really be rule one in the FORWARD chain after running them both. The SNAT rule is in another chain, so it could have gone anywhere. Also, I suggest you make sure the first rule above is correct for your system. If the untrusted network is 192.168.0.0/24, and the trusted network is 192.168.1.0/24, you may need the source (-s) to be 192.168.0.0/23 to cover both. Or, perhaps just drop the -s option and match on the inbound interface (-i).

I suggest that the best way to build a firewall is to walk through each chain to see where and how (or even if) a particular packet will be handled. Don't do as we've been doing here inserting rules seemingly willy-nilly. Build your chain on paper with all the rules in the correct order. Then you won't make mistakes. You can always check afterward to make sure the rules are as you think you wanted them with: **iptables -t <table> -L -nv**. The above rule with the -v included will show you how many packets and how many bytes are affected by this rule. If, after a week has gone by, you still have rules with 0 bytes affected by it, you might want to relook at that rule's position in the chain. But just because a rule has affected packets doesn't mean it's in the right place. It may have only affected half the packets that really should have been affected.

I'm waiting for someone to write "the killer app" for Netfilter, and that would be a utility that runs tests, analyzes the rules and allows you to move them around and test again. But until that day comes, you'll have to do it by hand.

Some of you may have noticed that I make heavy use of -i eth0, or -i ! eth0, but in general match an interface. Often, you can probably see that this isn't necessary because I've limited the source IP address or some other part of the packet header that pretty much ensures matching what we want. But I do this for a particular reason. I turn off rp_filters (reverse path filters). These tend to interfere with legitimate VPN packets. Besides, Linux's rp_filter is nowhere near as granular as iptables.

Protecting Your Data

It's very difficult to make sure someone on the inside isn't passing data through your firewall that shouldn't go out. And maybe some folks have a legitimate reason for passing company data out through the firewall. But let's assume for now that that's not the case. You want to stop certain data from leaving (or at least attempt to do so).

We can try to prevent certain data from leaving by marking that data, then looking for that mark using the string match. Here I suggest a policy of putting a string, such as "Copyright, foo.corp, not for publication" at the top of those files you don't want sent through the firewall. Then, on the inner firewall, or as a rule on eth2 (where eth0 is the Internet, eth1 is the untrusted LAN and eth2 is the trusted LAN) on your outer firewall, you might want something like this:

```
iptables -t filter -I FORWARD -i eth2 -m string
--string="Copyright, foo.corp, not for publication"
-j DROP
```

A few words about this particular solution. First, ensure you have the ipcontrack module loaded. This will defragment packets and result in a much higher likelihood of seeing the string. Second, don't expect this to catch everything. Particularly, if a file has been compressed, the phrase will not be recognizable as such. So this does have limitations.

It will work very nicely, however, if you are running an IIS server and want to drop packets with the string root.exe, for example. The rule might look like this:

```
iptables -t filter -I FORWARD -m string
--string=root.exe -j DROP
```

While it might be amusing to use the MIRROR target and turn the attack back on the attacker, this would be an ethically questionable thing to do.

You also have the PSD (port-scan detection) match if you're still subject to this kind of activity. I don't see so much port scanning anymore as I see script kiddies that have a particular tool; they aim it at my systems and fire. Usually it's an FTP attack designed to compromise an IIS server running FrontPage extensions. I see an FTP in, then lots of activity trying to create _vti_private files and the like. We can stop this with:

```
iptables -t filter -I FORWARD -i eth0 -p tcp
--dport 21 -m string
--string="_vti_private" -j REJECT
```

Obviously, if you're running a FrontPage server and folks aren't "publishing" to it (which uses port 80) but moving their sites via FTP, the above won't work.

Targets and Matches and More, Oh My!

This article has not touched on a large number of extensions and targets. Some of you with very specific routing requirements might want to look at the MARK target, with or without the realm match, to do some really funky routing tricks. This will require use of iproute2 in conjunction with iptables. This is a very powerful combination for ISPs or others with very specific routing and bandwidth-limiting requirements.

Others of you probably wanted to see some ULOG target examples or `iplimit` or `mport` examples. But these are very similar to other matches or targets and are handled in the same way. Often the help in the kernel configuration will show you enough of an iptables rule fragment to make use of these extensions.

Just remember, only ACCEPT, DROP or REJECT are final targets for a packet and stop iptables processing of a packet. The RETURN target only terminates a chain, but not iptables processing.

I also haven't touched on the MANGLE table. But this table works in the same manner as the mangle target in ipchains. Try it out if you're so inclined. You may find you won't be able to use the numeric (hex) targets but have to use the descriptive values. If you can't remember what they are, try:

```
iptables -j TOS -h
```

This trick also works if you need a list of the ICMP types because you want to handle a particular ICMP type with iptables (such as permitting pings, which will be dropped by a firewall with `-m state --state ESTABLISHED,RELATED`):

```
iptables -p icmp -h
```

Armed with the correct ICMP name, `echo-request`, you can permit pings:

```
iptables -t filter -I INPUT -i eth0 -p icmp  
--icmp-type echo-request -j ACCEPT
```

You also can rate-limit this using either `iplimit` or `limit` if you're concerned about this. But note that limiting pings doesn't limit the amount of traffic on your link, just the rate at which you'll respond to this traffic. Anyway, standard ping packets are so small and normally sent only once a second by any given host that they're barely noticeable as traffic.

Errors

Some common errors I've seen with iptables scripts include choosing an inappropriate interface for packets. This includes not selecting all interfaces that might be affected. Often `lo`, the localhost interface, is forgotten about on systems used as both firewall and host (usually a system used in a home). I've also seen outgoing packets using the MANGLE table, the OUTPUT chain or the SNAT target that have `-i <interface>` rather than `-o <interface>`.

Sometimes rules get so specific nothing matches them. Try the most general rule you can get away with, adding match extensions only as required. Just be careful where these rules are located in relation to other rules, so they're not picking up packets you don't want them to.

Ensure you're using the right case: ipchains uses lowercase for its built-in chains, but iptables uses uppercase. Targets are also uppercase. Almost everything else is lowercase. If you're using the short options (as I did in this article), the chain action (Insert, Append, Delete, etc.) uses uppercase.

Conclusion

Netfilter and iptables make an extremely powerful firewall. But to take advantage of it, you need to master the basic syntax as explained in my first article ("Taming the Wild Netfilter", published in the September 2001 issue of *LJ*), have an understanding of the modules and matches available to you and have an understanding of what a particular system can know about a packet. Armed with these three things, you can build highly complex, tailored firewall solutions for whatever problem you might have.

Know how to take advantage of new and experimental matches and targets (and always test them). You learned this in last month's article with the iptables build targets of pending-patches, most-of-pom and patch-o-matic. Testing by creating and sending specific packets to a firewall interface is beyond the scope of this particular article, but a number of utilities exist to assist you here (sendip or ipmagic come to mind).

Build the missing modules for the kernel (make sure they're selected).

Build your rule chains, more specific rules first followed by more general rules. If it helps you organize things, go ahead and build custom user chains that can be called from another chain. While this was not covered specifically in this article, it was addressed in the September 2001 article. Use everything at your disposal, including the LOG target to help you see if particular rules were applied and to which packets.

With just some basic knowledge, iptables are not difficult to use. Read some iptables scripts on the Internet. I don't recommend using them as they are; they almost certainly won't work for you without a lot of tweaking, but they will show you syntax, rules (from which you can grab fragments), thought processes, etc.



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

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

[Advanced search](#)

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

[Advanced search](#)

Zope Page Templates

Reuven M. Lerner

Issue #98, June 2002

Discover how ZPT attributes can work with HTML to create useful and compatible dynamic web sites.

In the early days of the Web, nearly everyone working on a web site was a programmer of some sort. You could be sure that every webmaster had installed his or her own web server, knew how to hand-code HTML and could write rudimentary CGI programs on their own.

Over time many editors, designers and other nonprogrammers became involved in the creation of web sites. Although it was often possible to teach these people HTML, it was HTML that was only good enough for static web pages.

Dynamically generated pages, which were then created almost exclusively by CGI programs, are another story altogether. After all, if the designer wants to change a site's background color on a static site, then he or she can simply modify the appropriate HTML files (or site-wide stylesheet, in today's world). But if such designs sit within a CGI program, the designer then must ask a programmer to make that change. This situation is bad for everyone; the designer cannot easily experiment with new ideas, the programmer is forced to make small and annoying changes and other programming work is put on hold while the programmer makes the changes.

For a number of years, the mainstream solution to this problem has been the use of templates, which mix HTML and a programming language. Perhaps the best-known commercial implementation of such templates is Microsoft's Active Server Pages (ASP), but Sun's JavaServer Pages (JSPs) is also quite popular. Open-source software developers have produced many high-quality template implementations of their own, including HTML::Mason (which works best with mod_perl), PHP (a programming language used for web templates) and ADP (available with AOLServer).

The idea behind such templates is quite simple: everything is assumed to be static HTML, except what is placed within a special set of brackets. When working with such templates, the designer is basically told, "You can modify everything that doesn't appear inside of <% and %>." And indeed, this often can work well.

But over time, the drawbacks of such templates become increasingly apparent. For starters, what happens when you want to loop through a number of items that you have retrieved from a database, displaying each item in a different background color depending on its content? In such a case, you cannot ask the designer to ignore the code because the code and HTML are so intertwined.

Zope, the application server that we have been looking at the last few months, tried to solve this problem using something they call DTML (Dynamic Template Markup Language). As we saw several months ago, DTML is a programming language with an HTML-like syntax that allows developers and designers alike to create dynamic pages. DTML is powerful, flexible and easy to understand (at least if you're a programmer).

But if you're not a programmer, then even DTML can be difficult to understand, particularly when it is retrieving results from a database (as we saw last month). Moreover, HTML editors don't know what to do with the DTML tags, which can lead to the inadvertent mangling of DTML pages.

For all of these reasons, Zope Corporation (formerly Digital Creations), which develops the open-source Zope application server, is now encouraging developers to look at Zope Page Templates (ZPT), a new approach to templates meant to solve most of these problems. This month we examine ZPT, looking at ways in which we can use it to create dynamic sites.

What Is ZPT?

ZPT takes advantage of three facts: HTML can be expressed in XML syntax, often known as XHTML; XML allows you to mix tags and attributes from different document definitions by using separate namespaces; and WYSIWYG HTML editors generally ignore (but preserve) attributes they do not recognize.

ZPT thus modifies the core HTML syntax by adding new attributes that sit in a separate namespace. Because these attributes are in a separate namespace, they are legal XML and XHTML. Because they are attributes and not tags, most HTML editors will ignore them, rendering the tag as if the attribute did not exist. And because web browsers ignore attributes that they do not understand, they can render ZPT pages without having to go through Zope.

In other words, ZPT makes it possible for a programmer to mock up a page so that a designer can then edit and view it using any tools they like. Of course, the dynamic elements of the template go into effect only when the template is viewed using Zope.

In HTML, each attribute has a name and a value, e.g., in `Reuven's site`, the attribute href had the following value: `http://www.lerner.co.il/`. The name and value are separated by an equal sign (=), and the value sits inside of double quotation marks. None of this changes when we work with ZPT, except that the attribute name is defined by TAL (Template Attribute Language). TAL defines a number of different possible attributes names, each of which directs Zope to modify the template in a different way when it is displayed.

TAL defines the attribute names, but what defines the attribute values? For that, we use TALES (TAL Expression Syntax). TALES defines a number of sources for data, including Python expressions and values from the surrounding Python namespace.

The combination of an attribute name from TAL (which tells Zope how to handle the surrounding tag) and an attribute value from TALES (which tells Zope what value to use in this TAL expression) gives us amazing flexibility in building page templates. In addition, the fact that TAL and TALES are published and have open-source specifications means that you can add to them if you have specific needs that they don't cover.

Some Simple Examples

Now that we've discussed the theory behind ZPT, let's look at some examples. The first example is the skeleton document that Zope creates when we start a new page template. To do this, go to the /manage URL on your Zope server, and choose page template from the Add product list in the upper right-hand corner. (If you don't see page template on the menu, it could be that ZPT is not an installed product. You may have to upgrade your copy of Zope to a more modern version in order to use ZPT.) As is usual in the Zope world, you will then be asked to enter an ID (i.e., a unique character string that will appear in the URL) for this new product. Enter a short name and click "Add and edit".

When you follow these instructions for a DTML document or method, you are brought to an editing screen that contains skeleton DTML. The same is true for a page template, except that the skeleton document obviously contains TAL and TALES expressions:

```
<html>
  <head>
    <title tal:content="template/title">
```

```

    The title</title>
</head>
<body>
  <h2><span tal:replace="here/title_or_id">
    content title or id</span>
    <span tal:condition="template/title"
      tal:replace="template/title">
        optional template id</span></h2>
  This is Page Template
  <em tal:content="template/id">template id</em>.
</body>
</html>

```

This document might be short, but it effectively displays TAL and TALES and how they might be used in actual documents. Here are the TAL attributes that this skeleton uses:

- The `tal:content` attribute replaces the contents of the tag with the value of the TALES expression. So in this example document, the words “The title” will be replaced by the value of the TALES expression `template/title`, which we will discuss below. The `<title>` and `</title>` tags, along with their attributes, are left alone, but the text between these tags is changed when the template is presented by Zope.
- The `tal:replace` attribute is similar to `tal:content`, except that it replaces the content and the surrounding tags. This is frequently used with the `` tag, which is largely used as a placeholder for such markup in any event. Thus everything from the first `` tag to the first `` tag, including the tags themselves, will be replaced by the value of the TALES expression `here/title_or_id`.
- The `tal:condition` attribute only displays its contents if its value is a true value, which is anything other than 0, the empty string, an empty list or the built-in ZPT “nothing” variable.

You might notice that the second `` tag contains two TAL attributes, `tal:condition` and `tal:replace`. When Zope encounters multiple TAL attributes in a given tag, it first evaluates definitions, then conditions, then repeat loops, then content and replace tags. (You cannot have both content and replace attributes in the same tag, because they are mutually exclusive.) In the second `` tag, Zope inserts the optional template ID only if the `template/title` TALES expression is true.

In addition to content, replace and condition, TAL defines three other attributes:

- `tal:repeat` allows us to loop over a list of items. If your template will display a list of search results, rows from a database or files in a folder, then you can loop over the contents and display the contents in a variety of ways.

- `tal:attributes` allows us to replace (or add to) an attribute of the enclosing tag. For example, we can set the `href` attribute of an anchor (`<a>`) tag at runtime by adding a `tal:attributes` attribute in that `<a>` tag. The value of the TALEX expression will be used to set that attribute when it is evaluated at runtime.
- `tal:define` allows you to set a new variable, which is then accessible from within enclosed tags. This variable value can be passed either along to other TAL tags or displayed.

TALES

Now that we have covered TAL, let's look at the TALEX expressions that we can assign to a TAL attribute.

The simplest sort of TALEX expression is a “path expression”, which describes a Zope object relative to the template, its container or the request. Path expressions are similar to URLs, except that they begin with a name rather than the root object `/`. The final element of the path expression typically will be a property that can be displayed (using `tal:content`), tested (using `tal:condition`) or iterated over (using `tal:repeat`).

The ZPT skeleton document uses four TALEX path expressions:

- `here/title_or_id`, which returns the value of the optional “title” property (if it exists) or the mandatory “id” property (if there is no title) to a `tal:replace` attribute.
- `template/title`, which returns the (optional) title of the current template to `tal:condition` followed by `tal:replace`.
- `template/id`, which returns the “id” property of the current template.

A TALEX path expression that begins with “template” refers to the template object itself, while one that begins with “here” refers to the object to which the template is being applied—it may be the template itself or another object entirely. Some other relative markers are “request” (the Zope HTTP request object), “repeat” (information about current `tal:repeat` loops), “options” (for the parameters passed to the template) and “container” (for the folder of the current object).

TALES path expressions are great but are not as flexible as we might sometimes need. TALEX thus allows you to return Python code by beginning your TALEX expression with “python:”. For example, we can include the result of a simple calculation in a template with the following code:

```
<p>2 + 2 = <span tal:replace="python:2+2">
    number</span></p>
```

There is also a “string:” TALEs prefix that indicates that the value should be treated as a text string. String expressions may contain interpolated variable values by prefixing the variable name with \$, as in Perl and shell scripts. You may optionally surround the variable name with curly braces ({ and }), as in \$ {x}.

Standard Look and Feel

What we have seen so far is very nice for dynamic content generation. But one of the best things about DTML (or any other sophisticated server-side macro language) is the ability to define menus in one document, headers in a second and footers in a third, and then for each page to import them as necessary.

One way to handle this situation is to create three separate templates (menu, header and footer) in the current folder, importing them with TAL expressions such as:

```
<span tal:replace="container/menu">  
  menu goes here</span>
```

TALES looks at the current template's container, retrieves the menu object (which happens to be a page template itself) and inserts its contents into the current document in place of the tag.

Another way to approach DTML's flexibility is with the use of macros. Macros are common in many programming languages and allow us to create functionality that expands at runtime. The ZPT macro language is called METAL, and like TAL and TALEs, it is defined and invoked within HTML attributes, placed in the “metal:” XML namespace. METAL macros can define “slots”, or parameters, into which parameter values can be bound. It's easy to imagine how you could create a macro that handles the overall site design, with each document fitting into the slot that this macro provides. Changing the macro definition would effectively change the design of the entire site.

Conclusion

When I first heard about ZPT, I was sure that it was yet another new way to create templates that are incompatible with other techniques and technologies. But over time, I have become convinced that ZPT is indeed a clever and elegant idea, and one that offers advantages to developers and designers alike. Although it is not a complete replacement for DTML, I believe that most of my DTML usage can now be replaced by a combination of TAL, TALEs and METAL. I look forward to seeing how these technologies improve over time and how they are integrated more fully into Zope in the coming months and years.

Resources

email: reuven@lerner.co.il

Reuven M. Lerner is a consultant specializing in web/database applications and open-source software. His book, *Core Perl*, was published in January by Prentice Hall. Reuven lives in Modi'in, Israel, with his wife and daughter.

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

[Advanced search](#)

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

[Advanced search](#)

Programming Life!

Marcel Gagné

Issue #98, June 2002

Here are a few games that are both addictive and educational for those starting to learn programming.

Programmers are like minor deities, François. They begin with nothing but an idea and access to a few random bits and bytes, then *voilà!* Out of the chaos, they create a new kind of life that takes form on our Linux systems. It is truly amazing, *mon ami*.

Yes, of course, François. You are quite right. It is a little intimidating as well. Ah, the power. The responsibility. The sheer pleasure of creating something from nothing. Not just intimidating, *mon ami*, but intoxicating. Speaking of intoxication, our guests will be here shortly. Tuscany sounds good tonight. Why don't you run down to the cellar and bring up the 1997 Vino Nobile di Montepulciano? *Vite*, François! They will be here any moment.

But they are here already. *Bon soir, mes amis!* It is an exceptional pleasure to see you here again at *Chez Marcel*, where fine Linux fare is always on the menu. Please sit. François is even now in the cellar bringing up today's wine. Before you arrived, we were talking about the joys of digital creation, programming, which you may have noticed is this issue's theme. The Linux systems you use here in the restaurant and at home are the product of numerous talented and passionate programmers, all of whom set out to contribute something to this community. Sometimes what they create is a kind of cybernetic life, creatures and robots of various incarnations. If you have ever sat on the sidelines thinking to yourself that you would also like to join this group of creators, then tonight's menu is for you. Ah, François, *merci*. Please pour for our guests. Be generous!

One way to learn something new is through games, and so it is with Java. IBM offers something called Robocode, a programming environment where you can

create Java robots that battle it out in an arena on your screen. Each battle goes through several rounds, and stats are gathered so that you can see how your favorite robot is doing. From this information, you can tweak various parameters and make your robot more efficient. Will stopping and turning every 30 steps instead of every 100 increase your chances of success? The way to find out is to make the changes, compile, run and watch the results. This is a great starter for the new Java programmer; it can also be addictive.

To get started with Robocode, you will need the Java developer's kit. You can get a free copy from java.sun.com/j2se; simply download the package you need. I used the RPM install on my system, but you also can get a GNU tar shell script for other Linux distributions. I should probably warn you about that installation—the binaries did not get installed in the standard bin directories. To continue, I had to change my PATH to reflect this:

```
export PATH=$PATH:/usr/java/jdk.1.3.1_02/jre/bin
```

Once you have Java installed, head on over to IBM alphaWorks for your copy of the Robocode package, located at robocode.alphaworks.ibm.com/home/home.html. You'll find the installation process quite painless, as there is really nothing to compile. Simply run the installation program with Java using the following command:

```
java -jar robocode-setup.jar
```

The first thing you'll encounter is a window that asks you to accept the license agreement. This will be followed by a suggestion for an installation directory. I accepted the default, a directory called robocode in my home directory. When the installation completes, you can start the program with this:

```
cd $HOME/robocode
./robocode.sh
```

When you start the program, you'll see the Robocode arena where robot tanks do battle. Click Battle on the menu and select New. A card will appear asking you to select robots for the coming battle. All these are prefixed with "sample.", and each has a certain set of characteristics preprogrammed. For example, start with sample.MyFirstRobot and sample.Spinbot. Before you go ahead and click the "Start battle" button, choose the battlefield tab. For my 1024 × 768 desktop, I chose a nice, unassuming 800 × 600 battlefield. Finally, there are some rules to follow, defined under the rules tab, including the gun cooling rate and inactivity time. Be aware of these, but for now accept the defaults. Ready? *Allons, mes amis*. Press Start Battle.

You get robots like Sitting Duck, which just sits there and waits to be blown up. You can also work with Crazy, a robot whose motion and firing pattern appears

mostly random. Each can be loaded and modified to behave in ways different from the original programming definitions.

If, like me, you were not entirely satisfied with the outcome of this battle, it is time to get coding and modify your robot. The first time you decide to modify your robot, Robocode will build Jikes, the compiler (it even tells you to go get a cup of coffee). *Oui, mes amis*, I was asking the same question myself. How old should a programmer be before he or she starts on the real java? For a look at the robot editor and a game in progress, have a look at Figure 1.



Figure 1. Editing Robots with Robocode

The easiest way to learn is to work with the sample robots, and then venture out on your own when you feel more comfortable. When the compiler is ready, you'll be presented with Robocode's development environment. Click File, Open and then choose "sample". You'll get a new list with all the sample robots (such as Ramfire). The code for each of these robots is well documented, so a new programmer can quickly grasp the concepts of what is happening. For instance, what should your robot do when it hits a wall:

```
/**
 * onHitWall: Handle collision with wall.
 */
public void onHitWall(HitWallEvent e)
{
    // Bounce off!
    reverseDirection();
}
```

The various methods for the robot class are a snap to catch on to, ahead(), back(), fire(), fireBullet(), donothing() and so on. You'll find documentation for the API in your installation directory under javadoc/index.html.

François, if you would be so kind as to refill our guests' glasses, I have another similar programming environment that I would like to share with them.

Andreas Agorander's DroidBattles works on a similar principle as does Robocode. Instead of using Java as its language, however, this program works in an assembler-like language. At this low level, it's as close as you can get to the heart of your software creation. The program itself is built using the 2.X Qt libraries, so those of you running KDE 2.X should have no trouble building the program. But, *mes amis*, before you build it, it must first come to you, or rather, you must come to it. The latest source is available at www.bluefire.nu/droidbattles/index.html.

The strange thing with this installation was that all the files in the archive were datestamped with a future date, which gave me a little grief when I got to the `./configure` step. To make things right, I used the touch command to stamp everything with the current date.

```
tar -xzvf droidbattles-1.0.4.tar.gz
cd droidbattles-1.0.4
./configure
make
make install
```

What makes DroidBattles fascinating is that you build the robot in code as well as in hardware, adding CPUs, memory, engines, plasma guns and scanners. When you start the program by typing **droidbattles &**, you'll get a control panel like the one in Figure 2. To do battle, you need to create and program at least one robot. Pressing the Bot-creator button at the top of the panel is the first step toward unleashing your mechanical warrior.



Figure 2. DroidBattle's Control Panel

You'll be presented with a kind of shopping list for your robot's physical attributes, as well as a window for code. For those who have never looked at assembler, the code will seem a little strange but not for long. These instructions are for the CPU itself—not the one in your computer but the one onboard your robot. The instructions aren't complicated, and a complete manual comes with DroidBattles; just click Documentation on the panel.

Incidentally, if you go ahead and add tons of hardware to your robot, you may be able to assemble it, but the system won't let you run it. "Why?" you ask. The config editor defines the parameters we must adhere to when building our robots. For instance, you define a maximum number of devices or memory, as well as the cost of these items. Furthermore, a robot has a maximum cost and each of those devices cost money. (Fear not, *mes amis*, this is virtual money.) That cost is determined by how good a particular device is. Plasma guns cost anywhere from \$50 to \$3,500. These parameters allow you to mix and match the various characteristics within a limited framework. Will you stand a better chance in battle with two plasma guns but only half the processing power? Make the changes, save your robot and find out.

Finally, when you are through, click Tests in the Bot-creator and choose "Quick battle". You need to select at least two robots to have a battle, but they can be the same ones. The battlefield appears (see Figure 3) and the game is on. Click Play and watch the fun. As the battle progresses, you can click Pause and then switch to single-step mode. When running in test mode, note the debug window that pops up and displays memory locations, registers and instructions as they are being executed.



Figure 3. The DroidBattle Battlefield

This is all very geeky and a great deal of fun. You might also want to watch Andreas' page, as he is in the process of setting up a DroidBattle server.

Well, *mes amis*, I fear that the time has gone by very quickly today. One of the most difficult aspects to starting down a road like programming is deciding what it is you would like to create. Programs like Robocode and DroidBattles set the stage for you and draw you into a tournament where the code seems more like play than work.

Thank you once again, *mes amis*. Your visit to *Chez Marcel* is always a pleasure. Relax. Finish your wine. Enjoy. Until next month. *A votre santé! Bon appétit!*

Resources

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

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

[Advanced search](#)

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

[Advanced search](#)

BestCrypt: Cross-Platform Filesystem Encryption

Mick Bauer

Issue #98, June 2002

This month Mick reviews BestCrypt, an open-source application that allows you to share encrypted volumes between Windows and Linux systems.

March 2002 was a bad month for advocates of personal cryptography tools. Network Associates officially dropped support for its PGP Desktop products, easily the most popular, mature and useful suite of end-user cryptographic tools in common use. As much as I hate to admit the superiority of any commercial product over free tools, Commercial PGP, while far from perfect, seemed to have the best chance of bringing strong cryptography to the masses. The world needs good crypto, specifically user-friendly good crypto with a time- and brainpower-saving GUI, and nobody benefits from PGP Desktop's demise, particularly in the absence of compelling alternatives.

None of which is meant to in any way impugn the admirable work of Werner Koch and the GnuPG team, on whom I've heaped heartfelt accolades in earlier Paranoid Penguin columns. In fact, let me heap some more on now: GnuPG rocks [see Paranoid Penguin in the September and October 2001 issues of *LJ*]. It's evolved into a stable and mature application in an astoundingly short period of time and has already taken its rightful place among other essential Linux tools that are part of nearly every mainstream distribution. Geeks love GnuPG, and you should too.

Unfortunately, in a GUI-centric world, GnuPG's various front ends need to realize much more of their potential before we can realistically hope to get nontechnical users to embrace GnuPG. And without accommodating nontechnical users in this way, we can forget about bringing strong crypto to the masses, even if it's free. GnuPG has this usability issue in common with Linux as a whole (uh-oh, here comes the hate mail).

Furthermore, GnuPG addresses only part of PGP Desktop's functionality. Whereas GnuPG does match PGP Desktop's abilities in e-mail and file

encryption, among other things, it doesn't do filesystem encryption, which was one of the very best things about PGP Desktop. PGPdisk (PGP's filesystem utility) made filesystem encryption simple, fast and transparent.

About the only thing it lacked was client software for Linux, which seriously impaired PGPdisk's usefulness on dual-boot systems. As a dual-boot laptop user, I always found this frustrating; any portable system *must* have filesystem encryption on all OSes it boots, period. Sure, I could set up an encrypted loopback filesystem on my Linux installation, but that isn't cross-platform either. It would be much better to share a single encrypted partition between both environments than to maintain two separate "vaults".

That brings us, albeit obliquely, to the subject of this month's column, which is actually about neither PGP nor GnuPG. It's about BestCrypt, a commercial but open-source application that allows you to share encrypted volumes between Windows and Linux systems, with all the transparency, simplicity and speed of PGPdisk.

Overview

BestCrypt is a filesystem encryption utility that allows you to create, mount and manage "containers" (encrypted volumes) on your computer that look and behave like any other mounted volume but are stored as encrypted files when not in use. This protects your sensitive data from computer thieves or anyone else who achieves unauthorized access to your system.

Because BestCrypt containers are ordinary files, they can be stored on removable media, archived, e-mailed as attachments and in general, manipulated like any other file. BestCrypt containers even can be placed on network shares and mounted by remote clients (though of course only one client may mount a given container at one time).

In addition, a BestCrypt container may be mounted by either the Linux or Windows version of BestCrypt; the same file format is used by both versions, with no loss of functionality in either direction.

Getting and Installing BestCrypt

BestCrypt can be downloaded from Jetico, Inc.'s web site in Finland at www.jetico.com/download.htm. It's a fast site, and BestCrypt is fairly compact—the Linux version is only 160K! The Windows versions are bigger, due no doubt to the fact that they're binary distributions, whereas the Linux version is distributed as source code. (For now I'm going to focus on the Linux version, but will talk about the Windows version shortly.)

Before you attempt to install BestCrypt, make sure that you've got the source code to your kernel installed under `/usr/src/linux`, where `/usr/src/linux` is either a symbolic link to or the actual root directory of your kernel source code. If you use a stock kernel from your distribution, simply install the corresponding kernel source package (just make sure the version is the same and that `/usr/src/linux` points to its root). If you've never built a kernel on your system, you then need to change your working directory to `/usr/src/linux` and execute these commands:

```
make mrproper
make menuconfig # configure the source to match
                 # your kernel
make dep
```

You don't actually need to build the kernel (unless you want to) by then doing **make bzImage modules modules_install**; the point is to build your kernel source's dependencies so that BestCrypt can compile additional kernel modules that match. (The first time I built BestCrypt on my SuSE 7.1 laptop, I forgot that I'd never compiled a kernel on that system, so BestCrypt wouldn't compile. Following the above procedure and then trying again did the trick, though.)

Installing BestCrypt from Source-RPM

Once your kernel source is in place and its dependencies built, you can build and install BestCrypt. If you use an RPM-based Linux distribution, get the source RPM (as of this writing the current one is `BestCrypt-1.0b-5.src.rpm`) and build it with the `--rebuild` flag:

```
rpm --rebuild ./BestCrypt-1.0b-5.src.rpm
```

This will build a binary package of BestCrypt in either `/usr/src/redhat/RPMS/i386` (on Red Hat systems) or `/usr/src/packages/RPMS/i386` (on SuSE and probably others too). You can then install that package like you would any other, for example:

```
rpm -Uvh /usr/src/packages/RPMS/i386/
BestCrypt-1.0b-5.i386.rpm
```

After BestCrypt's binaries and READMEs are in place, the RPM's post-installation script will load BestCrypt's kernel modules. You're now ready to use BestCrypt.

Installing BestCrypt from Source Tarball

If you use a non-RPM-based distribution such as Debian or Slackware, download the tarball instead of the source RPM (the most current one at the time of this writing is `BestCrypt-1.0b-5.tar.gz`). Unpack it in `/usr/src`, change your working directory to `/usr/src/bcrypt` and do a **make && make install**. If your

kernel source is set up correctly, BestCrypt should compile and install without errors.

The tarball's Makefile, however, isn't quite as sophisticated as the RPM installation scripts. You'll need to load BestCrypt's modules manually before using BestCrypt for the first time. The simple way to do this is with BestCrypt's startup script, e.g., `/etc/init.d/bcrypt start`.

BestCrypt's Documentation and Control Panel

In addition to BestCrypt itself, you should download the documentation tarball too. This contains a directory providing BestCrypt's documentation in the form of HTML pages (Figure 1).

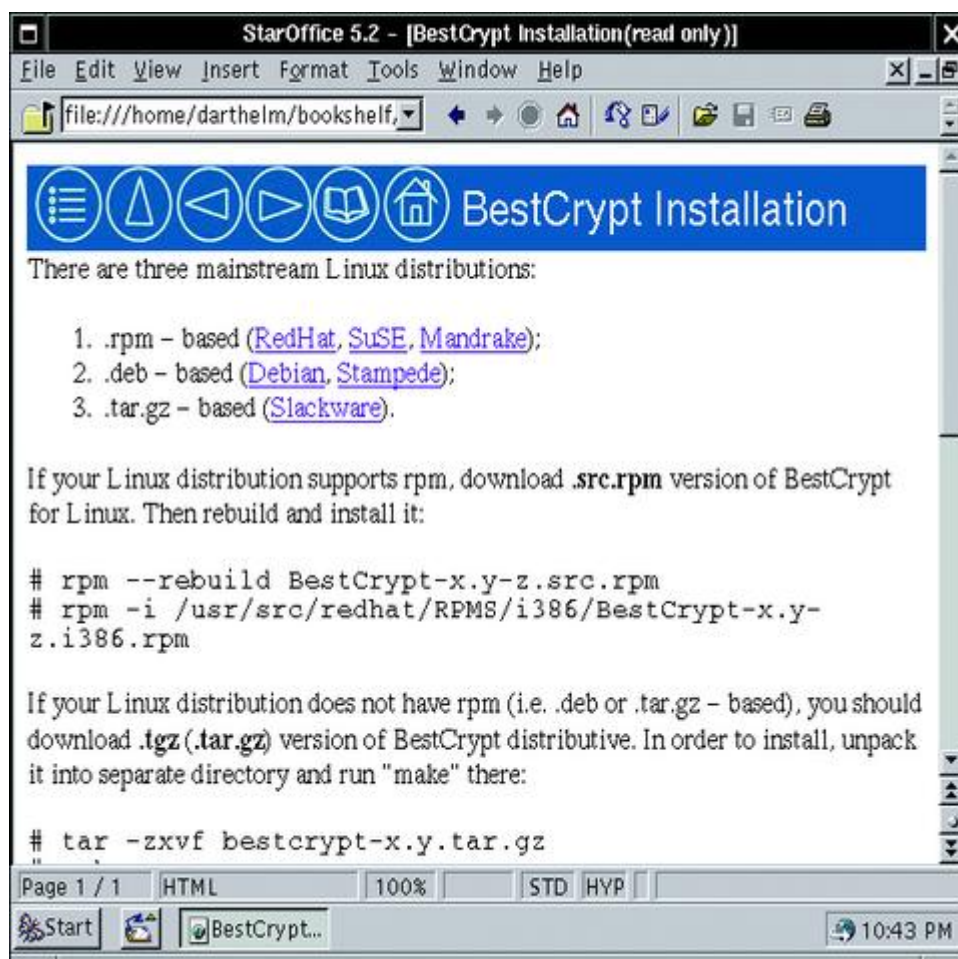


Figure 1. BestCrypt's Documentation

Another thing you may want is BC_Panel, the BestCrypt Control Panel. This is available only in the form of a binary RPM (though it may be installable under Debian using alien). BC_Panel provides a GUI for BestCrypt that very closely resembles BestCrypt's Windows GUI.

Since the current version of BC_Panel as of this writing is 0.2-1, and since it doesn't appear to support all the features of the command-line version (or of

the Windows GUI), BC_Panel appears to be a work-in-progress. Still, it's useful for some things and appears to be stable.

Using BestCrypt for Linux

Creating a BestCrypt container is quick and easy. Here is a sample session:

```
bctool new myvault.jbc -s 150M -a twofish
-d "my test vault"
Enter password:
Verify password:
```

BestCrypt has one main command-line tool, `bctool`. To create a new container you simply send `bctool` the command **new** followed by the container's filename, size, encryption algorithm and description. BestCrypt then prompts you for a password.

Make sure to use a strong password. Although all of BestCrypt's supported algorithms except DES use 128-bit or greater keys to encrypt containers, a container's key is itself hashed with your password. An easily guessed password means an easily decrypted container, no matter how big of a key it was encrypted with.

Make sure also to write down your password and keep it in a safe place, or choose a password you're positive you won't forget; according to Jetico, passwords are absolutely nonrecoverable, and there are no backdoors in BestCrypt for password recovery. This is very much a positive thing: while it means you will lose data irretrievably should you forget or lose your password, it also means the only way for an attacker to decrypt your container is to guess or brute-force your password.

After creating a container, you need to create a filesystem in it. This is done with `bctool`'s `format` command:

```
bctool format -t msdos ./myvault.jbc
```

Use the `-t` flag to specify a filesystem format supported by your system. If you're going to share this container with the Windows version of BestCrypt, be sure to specify `msdos` (if you actually use the `vfat` long filename extensions, aka Windows 95 long filenames, you should still format the container as `msdos`, and then mount it as `vfat`). BestCrypt can format containers in all file formats supported by your system.

To Root or Not to Root?

Once a BestCrypt container has been created and formatted, it can be mounted. The command syntax to do so is very similar to that of the familiar mount command:

```
bctool mount -t vfat ./myvault.jbc ./mnt/kraunj00lz
```

From this point on (until you dismount the volume) the volume may be accessed like any other directory. By default, the volume's user and group will be set to those of the user who mounted it, with permissions set to 0700 (**drwx-----**). In other words, other (non-root) users won't be able to access your volume unless you intentionally mount it with different ownership or permissions. You can specify a different user, group and permission mode at mount time with bctool's -o, -g and -m flags, respectively. See the bctool(8) man page for details and examples.

When you're done using the BestCrypt container, you can unmount it like this:

```
bctool umount ./mnt/kraunj00lz
```

While a BestCrypt container is unmounted, it can be backed up, copied and otherwise treated like any other file. While it's mounted, though, it can't be changed or manipulated (except by bctool).

BC_Panel: BestCrypt's Linux GUI

I mentioned that BestCrypt for Linux has a GUI, but as of this writing it's still in a beta state. BC_Panel (Figure 2) is available only as a binary RPM.

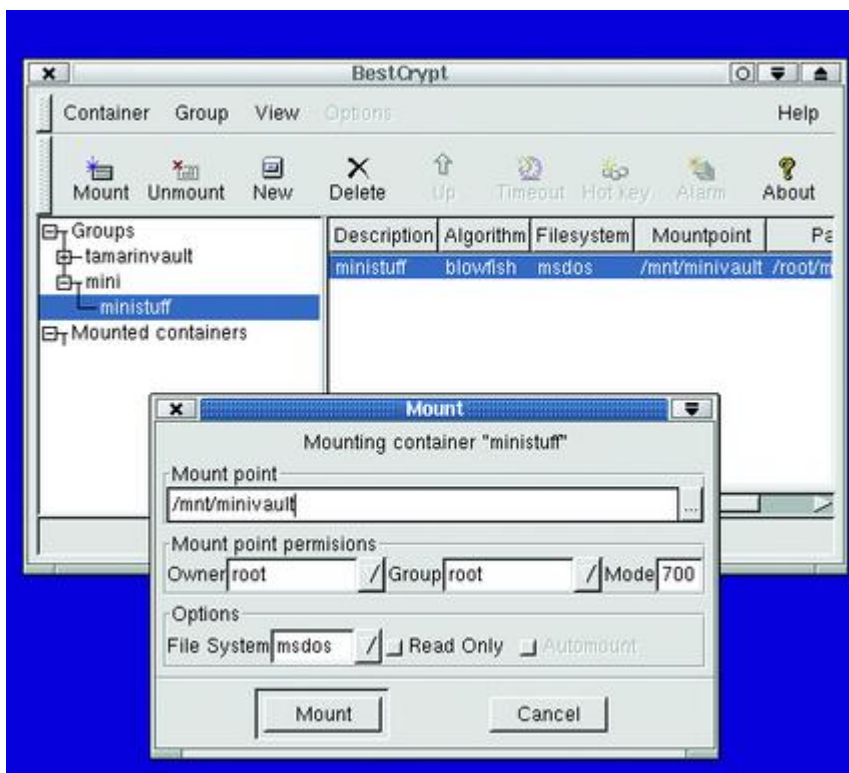


Figure 2. BC_Panel v0.2b, BestCrypt's Linux GUI

On the one hand, BC_Panel (the binary is actually called bestcrypt, but for clarity's sake I'll stick to BC_Panel here) is stable, obviously well written and at least cosmetically is very similar to its Windows counterpart. However, BC_Panel supports only a subset of the functionality provided by bctool and therefore, only a subset of the Windows GUI for BestCrypt; it isn't quite up to date with either.

For example, the New (container) dialog lists only a few possible algorithm options out of the ten or so actually supported. Worse still, trying to create a container with *any* of the algorithms presented as options by BC_Panel fails with an error message unless you're logged on (and running X) as root.

If you're root, BC_Panel does successfully create, format, mount, unmount and re-encrypt (with a different algorithm) BestCrypt containers, and even dynamically detects and lists containers mounted by the bctool command (for volumes on which the user running BC_Panel has read-permissions, that is). Thus, while I wouldn't rely on it for production use, BC_Panel seems to be useful for certain maintenance functions (if you don't mind conducting entire X sessions as root) and generally shows great promise; I hope Jetico releases a production-worthy version of it soon.

BestCrypt for Windows

Okay, I've established that BestCrypt is easy to install and use under Linux. But what about its compatibility with Windows? And what kind of potential does BestCrypt have for bringing disk-volume encryption to the Windows-using masses? The news is good on both counts.

For the past week I've been alternating booting my laptop system to Windows 98 and SuSE Linux 7.1, using the same BestCrypt container (which resides on my DOS/VFAT partition) as the working directory for my writing activities under both OSes. Other than an apparently harmless blue-screen error when I shut down Windows (Windows complains that one or more files are still open on the BestCrypt volume when it's unmounted), BestCrypt has performed flawlessly. I've lost no data, noticed no slowdown in disk performance when using the BestCrypt volume and have noticed no discrepancies whatsoever between the two versions of BestCrypt's handling of my shared container.

Equally nifty, I've had to invest practically no time at all in reading documentation or scanning mailing-list postings in order to use BestCrypt under Windows (unlike practically every other tool I've written about lately). BestCrypt's Windows GUI is extremely easy to use (Figure 3).

Granted, I'm intimately acquainted with the inner workings of public-key cryptography and have used other tools like PGPdisk for years. In other words, my credentials as an ordinary end user are suspect, to say the least. Still, I feel confident in stating that as far as usability is concerned, BestCrypt has at least an equally good chance as PGPdisk had in becoming the essential mainstream tool that elevates the masses to a Zen-like state of encrypted-volume enlightenment (and security).

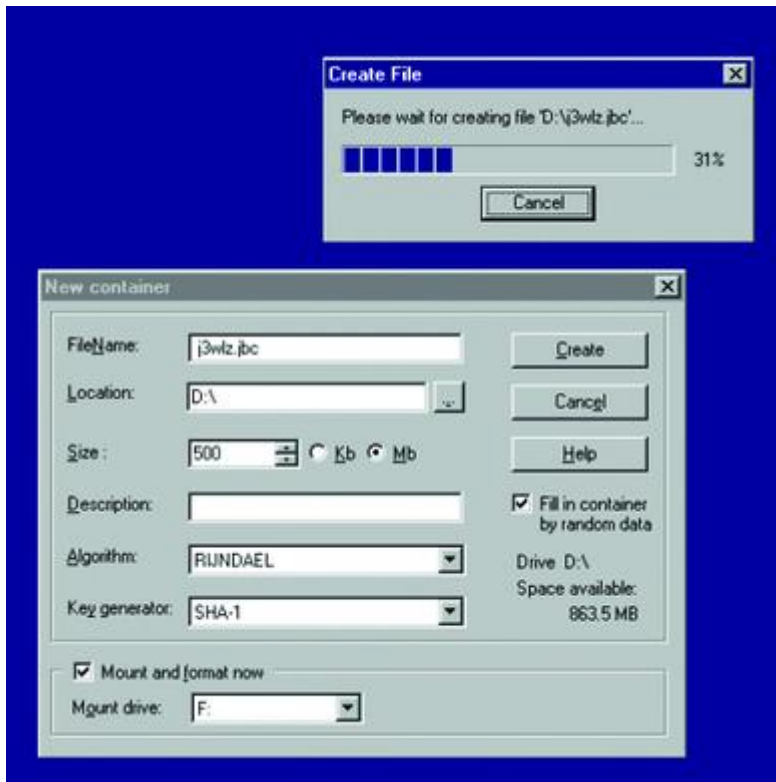


Figure 3. Creating a New BestCrypt Container under Windows

What I'm *not* confident stating is that based on painstaking cryptanalysis and code review, I believe BestCrypt to be impregnable. (Boy, I sure hope that sentence never gets partially quoted. Good thing those Jetico folks seem so highly principled!) Being neither a professional cryptologist nor even a programmer, I will have to leave it to others to judge the real strength of BestCrypt's security.

BestCrypt's Apparent Security

What I *can* tell you is that BestCrypt supports an impressive collection of known-good algorithms (or, as the more cynically minded might put it, it's "crypto-buzzword-compliant"), including the newly announced US Government Advanced Encryption Standard, Rijndael, plus two of the more promising runners-up in the AES contest: Ron Rivest's RC6 and Bruce Schneier's Twofish. If all three of those are too new for you, BestCrypt also supports Triple-DES, Blowfish (with several different key sizes), IDEA, CAST and the Russian Federal

standard GOST algorithm. BestCrypt also supports “single” DES, though its use isn't recommended due to its easily brute-forced (small) key size.

As an added bonus, Windows users get two additional features: Swap-file encryption, which protects you from attempts by others to extract passwords and other sensitive data from your Windows swap-file, and BCWipe, a low-level file eraser. Of these two, the swap-file encryption feature doesn't appear to be part of the Linux version yet.

BCWipe, however, can be purchased separately for Linux (i.e., it isn't bundled with BestCrypt as it is in the Windows version). BCWipe, like PGP's Wipe feature, repeatedly overwrites the data that remains when you “delete” a file, making it nearly impossible for deleted data to be recovered by any but the most sophisticated disk-recovery tools (if at all).

Thus, to the best of my qualifications in determining so, BestCrypt's security *appears* to be strong from a technical standpoint: it supports a number of important cryptographic and noncryptographic security technologies.

Conclusion

My evaluation of this product has centered on functionality, usability, Linux-friendliness, support for algorithms I like and trust, and of course, overall polish. I think BestCrypt succeeds on all those counts, and based on the consistently high quality of Jetico's software in those areas, I'm willing to believe that their cryptographic implementations are equally meticulous and well executed.

In conclusion, BestCrypt is an impressive product. If stability, comprehensive and modular support for a variety of popular encryption algorithms, and overall tightness are anything to go by, it seems to be highly secure as well. I enthusiastically recommend considering BestCrypt for your filesystem encryption needs, especially if you use both Linux and Windows. It's helped restore some of my hope for a more crypto-enabled populace and has been fun to play with, too.

[Product Information/The Good/The Bad](#)

Mick Bauer (mick@visi.com) is a network security consultant in the Twin Cities area. He's been a Linux devotee since 1995 and an OpenBSD zealot since 1997 and enjoys getting these cutting-edge OSes to run on obsolete junk.

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

[Advanced search](#)

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

[Advanced search](#)

Silicon Grail RAYZ

Robin Rowe

Issue #98, June 2002

This month Robin looks at how special effects are assembled with RAYZ in several new and upcoming motion pictures.

Last month we looked at Nothing Real's Shake, compositing software used on many major motion pictures to add special effects or to overlay animation. Nothing Real was recently acquired by Apple, which plans to use its fast scanline rendering technology in future Apple products. Shake's uncertain future on Linux may benefit competitor Silicon Grail's RAYZ.

Both RAYZ and Shake use an interactive scene-graph interface for defining how effects are assembled. While similar from a user-interface standpoint, RAYZ doesn't use the scanline rendering approach of Shake. "I think caching schemes are more important today than scanline algorithms for optimizing rendering", says Silicon Grail President Ray Feeny. "Memory is much cheaper now. Scanlines can actually be slower than keeping more blocks in RAM—for example, with rotations."

Hollywood effects company Flash Film Works used RAYZ on Linux for three recent Warner Bros. productions: the Arnold Schwarzenegger action film *Collateral Damage*, the Robin Williams dark comedy *Death to Smoochy* and the Eddie Murphy gangster sci-fi film *Pluto Nash* (to be released August 16, 2002). "RAYZ has the basic features a compositor should have, plus warp, morph and CineSpeed", says Flash Film Works Technical Supervisor Dan Novy. RAYZ is a new codebase. Silicon Grail's previous compositor, Chalice, didn't support Linux. "Silicon Grail held discussion groups and polled Chalice users over a year for power-user input", Novy says. "The RAYZ flipbook is just amazing. It can display 2k resolution in real time. Nodes will cache ahead as it is playing. I can work on the shot as it runs, do color-match changes in real time."



A simple color correction in RAYZ. Top-left is the flipbook preview player. Bottom-right panel is a tree-graph view of attached filters. Bottom-left panel contains settings for selected filter.

"In *Death to Smoochy* RAYZ did a lot of compositing of characters in cars and boats", notes Novy. "We used RAYZ to add muzzle flashes to guns and for street reflections. We did a lot of 3-D effects on top of 2-D plates." In one scene, the RAYZ warp and morph feature was used to repair a piece of a physical set that broke during a take. Although other takes had the set intact, the director felt that the take with the set problem had the best performance.

"For *Collateral Damage* we did a lot of retiming", says Novy. "Retiming 24fps into 36fps creates slow motion where none was shot." The RAYZ software creates tweens, extrapolating the missing frames. "Retiming may not be flawless, but it's better than double framing. REALVIZ ReTimer is an alternative, but that isn't built-in and would have to be rendered out." In *Collateral Damage*, a lot of compositing was done to place Schwarzenegger in front of explosions.



For *Collateral Damage*, Flash Film Works used RAYZ to composite actors working in front of a blue screen over background footage featuring an explosion, plus additional elements of explosions, debris and a falling pole were added.



For *Collateral Damage*--the helicopter on the left is real. The one on the right is a 3-D creation built in Maya and composited into the shot using RAYZ.

For *Pluto Nash*, RAYZ composited in moonscapes created in LightWave on Windows NT and blue-screened with Ultimatte. "The RAYZ net license Ultimatte feature is important to us. Digital Fusion, Combustion and After Effects require a dongle to use Ultimatte. You can't float to any machine like RAYZ." For a 3-D theme-park ride film, Flash Film Works is developing a plugin to distort the pixels to create parallax. A smoke plume can be adjusted to match the inter-ocular distance and create a composite for each eye. Their plugins are in Perl and C++. Novy is considering a switch from LightWave to Houdini, a popular 3-D package for Linux.

After Kodak stopped making the Cineon compositor, Silicon Grail acquired the technology and incorporated it into RAYZ as Grain/Degrain, Sharpen and CineSpeed. "A unique thing about RAYZ is it can keep data in Cineon 10-bit log

color format”, points out Novy. “Every other compositor makes you convert to 16-bit linear or to float. With RAYZ you can convert or keep it 10-bit all the way. It will do the internal math with logs.” Video monitors are 8-bit linear and need LUTs (colorspace lookup tables) to approximate film. “Because it was part of the Cineon technology incorporated into RAYZ, it can do full-monitor calibration to bring all our monitors into peak performance.”

“RAYZ runs on all of our operating systems and all our hardware”, notes Novy.

We're even using RAYZ on Alpha workstations, the 64-bit 21264. Back when we purchased these Alphas for *Deep Blue Sea* there was nothing faster, but a year and a half later they were in a corner doing nothing. Because of up-to-date Alpha support in RAYZ, we could resurrect this obsolete but fast hardware. We're moving toward Debian because Red Hat support for Alpha is lagging. I think Linux will dominate the motion picture industry like SGI did, but I am not sure about Red Hat.

Santa Monica-based Island Fever Productions is using RAYZ for visual effects in the 3-D, large-format film *Haunted Lighthouse* starring Lea Thompson. Due for summer 2002 release at Busch theme parks, the theaters showing the film will incorporate physical effects such as vibrating seats, water and odors.

“We're using RAYZ for all the compositing on this movie”, says Island Fever President and Visual Effects Supervisor Nick Ilyin. Compositing a large-format film shot in 65mm is especially demanding. The film is scanned at 4k resolution instead of the usual 2k typical with a 35mm film. “At 2k resolution performance is pretty reasonable, but at 4k RAYZ is handling four times the normal resolution of 35mm. Then, double that for 3-D. Our machines need a lot more memory to handle that. We use 2GB dual-processor workstations.” The 65mm scans are done at IMAGICA and delivered in Cineon DLT format at 4,096 × 1,536 resolution, stereoscopic. The finished 70mm 5-perf prints will be shown using two projectors and with Polaroid glasses for 3-D.

Island Fever Productions also uses Linux to create commercials, including three for Orkin showing cockroaches crawling on TV screens. Four commercials for Meijer Toy City feature funny animated penguins. “We created a penguin sculpture that was digitized and brought into Maya for 3-D character animation, then to Houdini for deformation and lighting and compositing, and then output to Pixar RenderMan”, says Ilyin. “We have written a lot of custom code to integrate that pipeline as well as a custom lighting package for Houdini.” Houdini's limited built-in compositor isn't powerful enough for projects like *Haunted Lighthouse*, which is why Island Fever recently moved to RAYZ for compositing.



For a Meijer Toy City commercial, 3-D penguins created by Island Fever Productions using Houdini on Linux. MPEG copies of these amusing commercials are on-line at www.island-fever.com/meijer.html.

Ilyin says Linux is definitely getting there, but wishes it were more dependable. He sometimes finds his renderfarm locked up, a problem that the industry generally attributes to issues being worked on in the Linux implementation of NFS. "We're hoping to have everything eventually in Linux, but support can be a problem. Red Hat support isn't enough." Island Fever uses Linux Red Hat 7.1 and 7.2, SGI IRIX and Windows 2000.

Drew Perttula is establishing a RAYZ community web site at rayz.bigasterisk.com to offer tips, examples, discussions and code (plugin code and external utilities). As a freelance effects compositor, Perttula has worked on several independent films. "I stumbled upon the beta version of RAYZ back in April 2001", says Perttula.

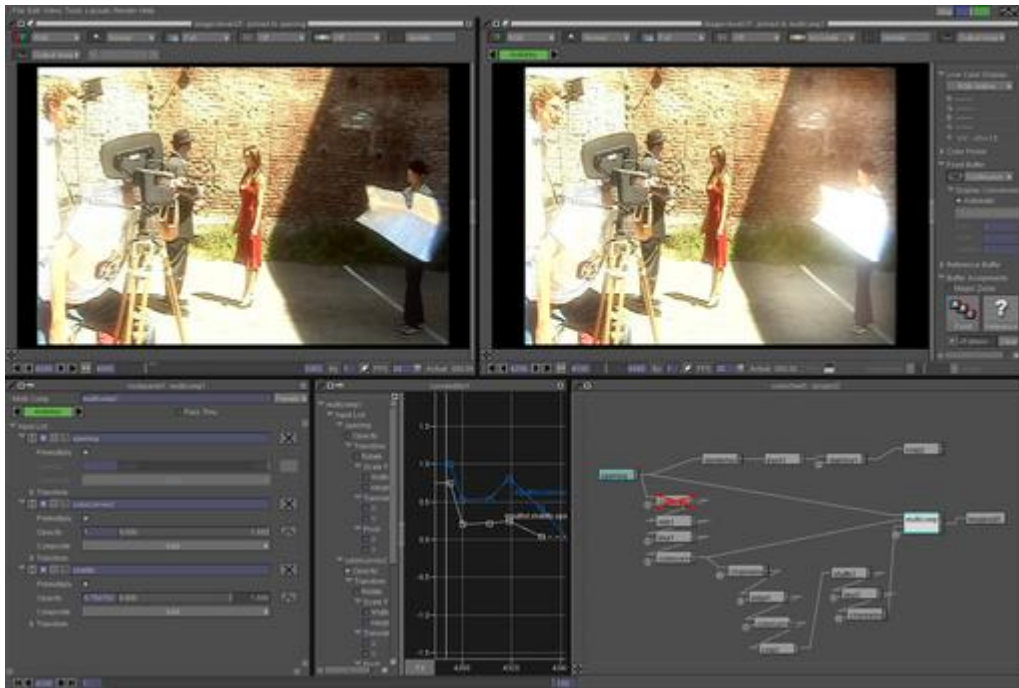
Since then I've been very active, with more than 600 feature and bug reports. Earlier versions of RAYZ seemed prematurely focused on optimization. Stability has gotten much better lately. In a few weeks the 2.2 version will be coming out with some exciting changes for plugin writers.

For indie film *Less Miserables*, Perttula is working on a background replacement for a scene shot on Mission Street in San Francisco. Where the street background has the wrong look, it is being adjusted to look more like Mission Street. "It's difficult because the actress' wardrobe presents very soft edges with feathers. Fortunately, I'm able to adapt a University of Washington friend's research to find edges, then import the result into RAYZ."



For *Less Miserables*, a background replacement made difficult by soft edges with feathers (before and after).

Less Miserables is a comedy about what can go wrong on a motion picture set. Director Justin Lomba was turned down by HBO's Project Greenlight but produced the film anyway on a \$20,000 budget. In one scene, the gag is that a grip shines a reflector directly into the lens of the camera, blinding the cameraman. "The reflector as shot is actually in a shadow, so we have to add the flare in post using RAYZ", explains Perttula. "In another shot we're hoping to get a friend at PDI to model a 3-D camera crane that I'll composite in to make it look like a bigger movie set."



For *Less Miserables*, lens flare effect added to reflector by RAYZ in a scene where a clumsy grip holding a reflector is supposed to be inadvertently blinding the camera operator.

Another shot for *Less Miserables* needs people added:

A lunchtime scene at a fountain in San Francisco should be filled with people, but the shot was actually done on a weekend when it was more deserted. If we had been called in earlier we might have simply had some friends walk in the background as extras. It is going to take some work to comp the crowds in afterward.

“Projects I worked on at PDI include indie *Memories of You*”, says Perttula.

People at PDI work on independent projects all the time, and I was brought in by my friend Marlon Montgomery. Director Shane Pollack wanted red rain and other supernatural effects, such as an actor crying a blue tear. That was fairly complex, with a couple hundred nodes in RAYZ and the rain from Maya and stock footage (from www.artbeats.com).



For

For his own project, *Golden Memories*, Perttula demonstrates a shot from a scene where a woman on vacation is feeding her fish by sending envelopes of fish food by mail. As it falls in the fish tank the envelope disintegrates. *Golden Memories* screened recently at ROBOTMEDIA, a monthly festival in Berkeley. "I'm taxing RAYZ by making each fragment of the envelope a block. I used Python and Pygame to create the blocks, so I don't need to manually create all the nodes in the RAYZ GUI."

Using Python with RAYZ

"RAYZ appeals to artists because it has a simpler interface than Shake", says Perttula.

And for programmers, RAYZ has a much better documented API than Shake. With its text-based file formats you can easily create little RAYZ accessories. RAYZ doesn't have a statistics gatherer but I built one in ten minutes. I'm using Perl and Graphviz to create a pretty shot tree diagram of what went into RAYZ and what came out.

Perttula says that the ease of having Perl create an overview of how a whole project works is a testament to how easy it is to read RAYZ files and do interesting things with them.

The Mouse that Roared

To get RAYZ, download the license manager (license_manager.linux.intel.tar.gz) and the application (rayz-2.0.i386.rpm) separately from the Silicon Grail web site. Using alien we converted the RPM to install on our Debian system. The alien tool handles conversions between Red Hat RPM, Debian deb, Stampede slp and Slackware tgz formats.

```
alien -k rayz-2.0.i386.rpm
dpkg-deb -c rayz_2.0-20020129_i386.deb
dpkg -i rayz_2.0-20020129_i386.deb
cd /usr
tar xvfz /install/compositor/rayz/z/
      license_manager.linux.intel.tar.gz
grail/hostinfo
grail/Install
grail/grailadmin -r localhost
grail/graild -l /var/log -k /usr/grail
rayz
```

Using **dpkg-deb -c** reveals where files will be placed without extracting them. Many Linux programs fail to mention how to launch them, but dpkg-deb makes that easy to figure out. (Using dpkg-deb -Xe is an option if you ever want to simply extract a deb as though it were a tar file without installing it.) Since we see that the main directory is /usr/grail, we chose to install the license manager there, too. The hostinfo command displays your machine's ID number to send to Silicon Grail to get a license key. Rather than installing the license manager as a daemon, we ran it manually.

The RAYZ development team consists of five developers, all working on Linux workstations. "We made a strategy decision three years ago to focus on Linux", says RAYZ Product Manager Craig Zerouni. "It was clear to us then that Linux would be the place to be." Windows and SGI IRIX versions of RAYZ are built from a single Linux-developed codebase of about a million lines of code. Tools used include gcc, g++, CVS and Jam. "We recently switched from **make** to Jam", says Zerouni. "Our time for a recompile dropped from hours to minutes because Jam is smarter than make depends." Head of R&D Kimball Thurston adds, "We can't use autoconf or tools like that cross-platform, but Jam works the same across all platforms." Using Jam requires rewriting Makefiles into jamfiles. A familiar project that uses Jam is FreeType.

At \$9,900 US per GUI and \$2,900 US per render thread, RAYZ isn't cheap. However, Silicon Grail is cooperative with plugin developers and university users. Silicon Grail welcomes developers to become third-party plugin providers. "You don't have to be a studio to be involved with RAYZ", says Zerouni.

Our plugin SDK, documentation and examples are included with the evaluation download. You can use all of RAYZ internal operations (for example, image

rotation) as building blocks for your own code. If you just want to develop we can set up a three-month trial license beyond the one-month evaluation license available with the download. Depending on what you have in three months that may be extended.

Companies such as Martian Labs are creating RAYZ plugins, as well as The Foundry in the UK with Keylight and Photron with Primatte.

Resources

email: Robin.Rowe@MovieEditor.com

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

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

[Advanced search](#)

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

[Advanced search](#)

Striking a Nerve

David A. Bandel

Issue #98, June 2002

Readers respond to David's recent topics of interest, frustration and spam.

Recently it seems I hit on three very hot topics (all at once, too), that is if the volume of mail I've received is any indicator. The first of these topics can be reduced to library versioning problems on distributions and my need to keep several kernels of varying levels running to make everything I wanted to compile and run do so. Guess I'm not the only one. Perhaps a plea to programmers not to use the latest bleeding-edge library version would help, and this would be the *LJ* issue for that. So programmers, if you're listening, how about helping us users out!

The second issue that struck home was spam. We've seen the fall of ORBS followed closely by ORBZ. A number of other blacklists have sprung up in their place like weeds, but without a track record, just good intentions. The Razor database is suspect, with Razor sidelining several mailing-list messages that were obviously not spam. So lists and databases only work so well. I've looked over yet another antispam package that seems promising and is tunable. Let's see how that turns out (see SpamAssassin below).

Finally, a lot of folks are eager to dump Windows but can't seem to replace that Quicken package. Financial packages are dull, uninteresting and few programmers willingly commit the programming career equivalent of hara-kiri by writing one of these packages. But they are needed. One package with some promise has gone from open- to closed-source. I don't review commercial packages, but those of you needing a personal financial package might take a look at MoneyDance on the AppGen site (www.appgen.com).

Mail::SpamAssassin www.spamassassin.org

I've looked over a very large number of spam filters, and not one is perfect. It seems some folks have been poisoning the well where Vipul's Razor database is

concerned, so I tried SpamAssassin, which has a setup similar to Vipul's Razor. One thing about SpamAssassin, it's easily reconfigured. During my test period, I had several hundred spam messages identified. I only had one spam at 4.6 make it through and one friendly message at 5.5 get sidelined. But SpamAssassin allows you to create white and black lists. So if you have friends in, say, Costa Rica who use acr.co.cr in their e-mail addresses, you can whitelist only their specific address, while the spam kings using acr.co.cr are summarily sent to /dev/null. Excellent. Requires: Perl, Perl modules Net::DNS, Mail::Internet, Net::SMTP and procmail.

xlog people.debian.org/~pa3aba/xlog.html

For all you ham operators out there, this is a great logging tool for your contacts. You can have multiple logs. You can add and delete bands in Preferences. The date is filled out, and on contact you just click the Time button and the time is filled in. Fill in the calling/responding station, add some remarks, select the band and click Add. You can search the log and more. It has an extremely user-friendly interface—heck, even a non-ham could work this log! And I should know. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.

Remote Accounts Handler www.entropika.net/racs

This particular Bash script goes a little beyond gasman. Not only does it store a list of your remote accounts and logins in GPG-encrypted form, it also allows you to connect to them by calling Racs with the alias for your remote account as the argument. It then fires up the application (SSH, Telnet, FTP, SFTP, HTTP, MySQL) and connects you to the account. Requires: Bash, expect, dialog (optional), GPG.

Penetrator www.triptico.com/software/penetrator.html

This Perl application will index all your files so you can perform a word search à la htDig or another search engine. But it also works locally on your hard disk and anywhere you have read privileges. So if, like me, you have years' worth of text-type documents and would like a word index of them, check out Penetrator. Its first run may take awhile, but after that, adding entries are quick and easy. If you take advantage of the optional SQL capabilities, you can perform SQL searches on the database without Penetrator's help. Requires: Perl, Perl modules DB_File, Getopt::Long, DBI::Pg (optional).

dnotify www.student.lu.se/~nbi98oli

This utility sleeps in the background until a directory it is watching has a file accessed or changed in a predetermined way; it will then perform the specified command. This utility could be of particular value as part of an intrusion detection system. Find a rootkit? Let dnotify send you a message when the directory containing the file has been accessed. Requires: glibc.

yesClock www.germane-software.com/software/yesClock

Here's a different idea in a clock. It will tell you the time, but also shows you (provided you've set your preferences to the appropriate lat/long) your relative day/night position. This one is just for fun. Requires: JVM2.

Nessus www.nessus.org

Three years ago I reviewed Ted, an excellent RTF word processor, Nessus, a security check program and Nmap, a network scanner. A tough choice, but I went for Nessus.

Okay, I cheated a little. Nessus uses Nmap as part of its routine. Nessus is probably the most complete and powerful security auditing tool available at any price, and this one's free. If you use the development release, you'll get a good look at all your vulnerabilities so you can do something about them. If you are responsible for network security, this package is a must-have. Requires: libX11, libXext, libXi, glibc, libdl, libgdk, libglib, libgmp2, libgtk, libm, libnsl, libresolv.

Until next month.

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

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

[Advanced search](#)

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

[Advanced search](#)

Linux at the Embedded Systems Conference 2002

Rick Lehrbaum

Issue #98, June 2002

At ESC in San Francisco, Rick finds that embedded Linux isn't "news" anymore.

The Embedded Systems Conference (ESC) held in San Francisco during the week of March 11, 2002 was expected to draw over 15,000 attendees from all facets of the embedded systems market. And, despite the much slower than expected economic conditions, event organizers reported that attendance reached around 12,000. Even with the attenuated turnout, this year's ESC product and technology expo was the largest ever, occupying two large halls in the sprawling Moscone Convention Center complex.

From the embedded Linux perspective, the really *big* news this year is that embedded Linux isn't news anymore! Embedded Linux is now universally counted among the top three "must-have" embedded OSes, which virtually all embedded hardware (chips, boards, systems), middleware, applications and tools must support. The other two are generally VxWorks and one or another flavor of Windows Embedded. Beyond embedded Linux, VxWorks and Windows Embedded, it now appears that all the other embedded OSes have been lumped into the nondescript category of "other" and are supported only when a major customer is willing to provide the funding or purchase commitment.

Here, then, are some highlights from ESC 2002 San Francisco, from the embedded Linux perspective.

Embedded Linux Consortium Kicks Off Embedded Linux Platform Standard

The Embedded Linux Consortium held an open technical meeting on March 12, during which the two-year-old group moved beyond its initial role of primarily evangelizing the adoption of embedded Linux to one of starting to create a unified embedded Linux platform specification (ELC Platform Spec). There were over 125 attendees, and although no sign-in sheet was circulated, the RSVPs reportedly included representatives of some of the world's largest and most

influential software, semiconductor and electronics companies, including HP, Hitachi, IBM, Intel, Microsoft, Motorola, Panasonic, Samsung, Sharp, Sony, Texas Instruments, Toshiba and Wind River.

Some observers have complained that the process of moving the ELC Platform Spec forward is going far too slowly to be useful to the market, while others are glad that a proper foundation (the Intellectual Property Agreement, or IPA) is now in place, allowing large corporations to participate fully in the spec development process. In any case, the ELC's initial bylaws prohibited standards development activities, so a formal document was certainly necessary to enable this important change to the group's charter.

The ELC meeting's two keynote presentations are available on-line:

- "The Future of Embedded Linux (and the evolving role of the Embedded Linux Consortium)" by Rick Lehrbaum, Founder and Executive Editor, LinuxDevices.com: www.linuxdevices.com/files/article026/index.html.
- "Update on Embedded Linux in Japan (and EMBLIX, the Japan Embedded Linux Consortium)" by Dr. Tatsuo Nakajima, chairman of the Japan Embedded Linux Consortium (EMBLIX): www.linuxdevices.com/files/article025/index.html.

The ELC has now formed its first working group, chartered to develop the new ELC Platform Spec, and is encouraging volunteers to join the efforts of finalizing the spec with an eye toward a 1.0 release by the end of 2002. Proposals for other working groups, having orientations such as real-time Linux, security, wireless APIs, high availability, device drivers, etc., have also been solicited by the ELC (www.embedded-linux.org).

A Quick Tour of the ESC Vendor Expo

A total of 39 companies listed themselves under the category of embedded Linux in the ESC show guide. Here is a sampling of some of the many embedded Linux-oriented demonstrations, technologies and products that I found at the show.

- Arizona Cooperative Power (www.azpower.com): (in Hitachi's booth) Rob Wehrli demonstrated μ Clinux running on a new Hitachi H8S-2148AF development platform. Wehrli says the board is "especially designed for μ Clinux...and is intended to make it easy for educators, students and companies to explore μ Clinux running on the H8S without incurring a hefty equipment expense." It is based on a 33MHz H8S-2674R and includes 8MB of SDRAM, 4MB of Flash and a 10Mbits/s Ethernet port, and will be sold for \$199 US.

- China MobileSoft (www.chinamobilesoft.com): made their first appearance in North America at ESC 2002, where they demonstrated mLinux, their embedded Linux distribution. They also showcased their embedded Linux GUI (mGUI) and microbrowser (mBrowser). Prior to the recent opening of a sales office in the US, MobileSoft's business has been focused on manufacturers of smartphones, PDAs and set-top boxes in China and Taiwan.
- Lineo (www.lineo.com): held a press conference to announce several new partnerships and strategies associated with the set-top box, residential gateway and handheld computer markets. One interesting announcement was of a new Linux-based PDA (the Kaii, mentioned briefly in last month's column), which is being developed in India and has a software stack similar to that of the Sharp Zaurus. Lineo explained that they are in the process of repositioning their products and resources around three key markets: smart handhelds, digital media and residential gateways. This theme was reflected in the company's ESC booth demos, which were much more oriented toward application-specific solutions than generalized tools.
- LynuxWorks (www.lynuxworks.com): announced the release of LynxOS version 4.0, which is said to feature Linux ABI compatibility. This means that LynxOS, a proprietary real-time operating system (RTOS) that previously offered a high degree of API compatibility with Linux (allowing it to run certain Linux applications after recompilation), should now be able to run some unmodified Linux binaries. Linux programs demonstrated running (unmodified) on LynxOS included the Opera browser and *Quake*. Note that this capability apparently requires that the Linux applications use dynamically linked libraries, since special versions of glibc library functions are required. LynuxWorks calls such programs "well-behaved applications" and points out that the LSB requires dynamic linking of libraries.
- Microtronix (www.microtronix.com): (in Altera's booth) demonstrated a Linux-based board support package for Altera's ARM-based Excalibur development kit, which allows development of multiprocessor systems combining Altera's Excalibur hard-core "stripe" along with one or more of Altera's Nios soft-core embedded processors in the programmable logic portion of the device. The Microtronix support includes a multiprocessing mezzanine board with multiple expansion headers to support Ethernet and display adapter cards.
- MontaVista Software (www.mvista.com): had their usual large pavilion with numerous hardware/software demos, including the MontaVista High Availability Framework, the Visual Age Micro Edition Java VM equipped with a Java AWT based on Qt/Embedded, a digital set-top box reference design based on IBM's PowerPC 405GP processor, MontaVista Linux

running on the Sharp Zaurus PDA and a demo of the latest real-time features (and supported processors) of MontaVista Linux version 2.1. One really cool demo was of a real-live Linux-powered soft-drink vending machine (the USA Technologies ePort), powered by a RadiSys StrongARM-based embedded SBC running MontaVista Linux. MontaVista also announced the receipt of an equity investment from Panasonic Digital Concepts Center (PDCC), a subsidiary of Matsushita Electric Industrial Co., Ltd.

- Red Hat (www.redhat.com/embedded): demonstrated their embedded software family (Embedded Linux, eCos, RedBoot, GNUPro tools) and showcased some interesting design wins, including a Brother printer (based on eCos), Symbol Wireless barcode scanner (Linux), Rymic ruggedized vehicle computer (Linux), Ericsson's Screenphone (Linux), Sony's PS2 (GNUPro), Intel's residential router (Linux), Iomega's Hip Zip (eCos) and Delphi's auto telematics system (eCos). One particularly interesting demo involved the results of a recent investigation by Red Hat's Clark Williams, which compares two popular ways to improve Linux kernel preemption latency: the preemption patch (pioneered by MontaVista) and the low-latency patch (pioneered by Ingo Molnar). Which performs best? Both, as indicated by a detailed whitepaper available online at www.linuxdevices.com/articles/AT8906594941.
- REDSonic (www.redsonic.com): showed the capabilities of RED-Builder, their GUI-based embedded Linux Image Creator and Integrated Development Environment (IDE), demonstrating the point-and-click simplicity of using it to build, download and debug complete, small footprint Linux systems for a wide range of embedded targets. Target platforms demonstrated included Ampro's EnCore 500 (x86), ITE 8152EVB (StrongARM) and MIPS 32-bit Malta Board. REDSonic also demonstrated SecureSOHO, a software stack for resource-constrained, Linux-based gateway/firewall devices, which provides extensive networking and security features along with an easy-to-use end-user configuration/control interface.
- RidgeRun (www.ridgerun.com): (in Texas Instruments' booth) demonstrated DSPLinux, an embedded Linux distribution and toolkit for TI's dual-core (RISC/DSP) system-on-chip processors. At ESC, RidgeRun announced that they have added dynamic linking support to μ Clibc. Previously, developers of systems based on MMU-less processors (e.g., ARM7) had hesitated to use embedded Linux due to concerns about the ability to keep their application code proprietary for various reasons, such as partner licensing agreements or protecting trade secrets. Dynamically linking to the LGPL-licensed μ Clibc libraries eliminates that problem. The new μ Clibc dynamic link support is being made available as part of a new board support package for TI's TMS320VC5471 dual-core processor.

- Vitals System (www.vitals.co.kr): (in Samsung's booth) demonstrated its family of vLinux Developer Packages, which are reference designs for Samsung's ARM7- and ARM9-based system-on-chip processors. The platforms include full source code and schematics, are sold for a one-time fee (without royalties) and are meant to provide turnkey designs for wireless LAN access points, SOHO routers and wireless LAN ADSL modem routers.
- TimeSys (www.timesys.com): despite the stormy economy, rode a wave of good news to ESC, announcing the receipt of \$15.5 million funding and issuing a press release declaring “Pioneer of Embedded Systems Software Poised to Assume Industry Leadership Role”. Their booth featured four demos that showcased their board support packages (BSPs), which include support for the TimeSys Linux/GPL Embedded Linux distro and associated toolchains, Windows cross-hosting support, plus add-ons for real-time performance, quality-of-service CPU reservations and real-time networking, running on a variety of processor architectures (x86, ARM, MIPS, PowerPC, UltraSPARC, XScale, SuperH) and SBCs. The company also announced several new BSPs and promised a steady stream of additional ones on an ongoing basis.

Elsewhere in the World of Embedded Linux

A new book on embedded Linux has been released by Addison-Wesley. *Embedded Linux: Hardware, Software and Interfacing* by Dr. Craig Hollabaugh is a 432-page book that teaches the development and implementation of interfacing applications on an embedded Linux platform. Hollabaugh says his book

documents “Project Trailblazer”, a hypothetical winter resort automation project—from initial funding, to design and then through implementation and system integration. You will follow the Trailblazer engineers as they select target hardware, create a development environment, interface various data acquisition, control, and multimedia devices, and then write device drivers and integration code.

Lineo tightens its belt (again). Citing “the impact of the economic downturn”, Lineo further reduced its headcount in March, from 138 employees to “between 75 and 80”, according to CEO Matt Harris. Harris says Lineo is continuing the process begun last fall of narrowing its focus to three key markets: handheld devices (PDAs and smart phones), edge devices (residential gateways, firewalls, routers) and digital TV (set-top boxes, entertainment systems). Last September, Lineo announced that it was laying off 60 employees and “spinning out” an additional 100, which was to leave the company with roughly 110 employees. A previous layoff, in June 2001, had reduced their

workforce from 322 to 280. Harris expects the latest changes to bring about profitability by mid-2002.

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

email: rick@linuxdevices.com

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

[Advanced search](#)

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

[Advanced search](#)

Bad Law

Lawrence Rosen

Issue #98, June 2002

Why UCITA is unacceptable to the Open Source community.

About a year ago I was working with another attorney to change the Maryland UCITA statute to be friendlier to open source. UCITA, the Uniform Computer Information Transactions Act, is a model code intended to be adopted by all states so that there is uniformity to the law of software licensing. UCITA provides default rules that apply when a software license omits essential terms. Another purpose of UCITA is to define what license terms are against public policy and thus cannot be enforced even if they are included in a license.

In many respects UCITA is like the Uniform Commercial Code. For example, the UCC sets rules to prevent a merchant from foisting shoddy goods on an unsuspecting public. In that same vein, early versions of UCITA stated that it is against public policy for a software license to disclaim the implied warranties of merchantability and fitness for a particular purpose.

UCITA requires, regardless of what the license says, that a licensor provide warranty protection if the licensed software turns out not to be fit for the ordinary purposes for which it was intended or to conform to the promises made in its documentation or marketing materials.

Damages for breach of warranty can be substantial. In appropriate situations, a licensee can recover for any loss resulting from the breach, the difference between the value of the software accepted and the software delivered, and even incidental and consequential damages.

The makers and distributors of open-source software cannot afford these implied warranties. If software is given away along with the source code, then how does an open-source licensor recover the cost of the warranties? That's why all open-source licenses declare that the software is made available *as is* and without warranty.

Maryland is one of the few states that has adopted UCITA. The implied warranties in the Maryland law were unacceptable to the Open Source community. After hearing our arguments, and to correct this situation, the Maryland legislature adopted the following amendment to UCITA:

The warranty [of merchantability and fitness for a particular purpose] does not apply to a computer program if there is no charge for (1) the source code, (2) making copies, or for use of those copies, (3) modifying, and (4) redistributing the computer program.

UCITA is adopted state-by-state. To help avoid dealing with this issue piecemeal, representatives of the Open Source community then sought the adoption of an amendment to the uniform code itself. The Maryland statute was submitted for consideration by the National Conference of Commissioners on Uniform State Laws (NCCUSL), the author of UCITA.

Instead, the provision the NCCUSL committee adopted reads as follows:

(a) Except as provided in subsection (b), the warranties [of merchantability and fitness for a specific purpose] do not apply to a computer program if the licensor makes a copy of the program available to the licensee in a transaction in which there is no contract fee for the right to use, make copies of, modify, or distribute copies of the program.

(b) Subsection (a) does not apply if the copy of the computer program is contained in and sold or leased as part of goods or if the transaction is with a consumer licensee that is not a software developer.

The replacement of “and” with “or” toward the end of subsection (a), and the omission in that sentence of the requirement that the source code be available, are very important changes. It means that companies that bundle their “free” software with software for which they charge license fees—as Microsoft does with Internet Explorer, for example—are eligible for the warranty exemption even though they do not satisfy any of the other criteria of open-source software. It guts the entire purpose of the amendment.

The addition of subsection (b) is another dangerous trap for the unwary. The second part of that subsection means that the warranty exemption is fine when the software is distributed to other software developers, but as soon as the software is distributed to real users or customers, the implied warranties are required. Thanks, but no thanks!

A letter from the National Association of Attorneys General, signed by attorneys general from 32 states, was submitted to the UCITA Standby Committee on November 13, 2001. It contains criticisms of UCITA in general, but contains no substantive proposals for any amendments of any kind. The letter effectively states that there are no conceivable amendments of any kind that might be proposed to improve UCITA as suitable law for computer information contracts.

As long as UCITA doesn't adequately address concerns about its fairness and effectiveness, including the concerns of the Open Source community, it will not likely be adopted by enough states to make it useful. We must remain vigilant, state-by-state, to prevent the adoption of this flawed law.

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

email: lrosen@rosenlaw.com

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

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

[Advanced search](#)

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

[Advanced search](#)

Identity from the Inside Out

Doc Searls

Issue #98, June 2002

Doc introduces PingID, an identity framework controlled by the individual rather than by any one company.

Back in 1995 or so, when my wife started using a browser, she came up with a wish list of features that nobody has come close to filling over the seven years since. One was a shopping cart that would persist from one site to the next. Another was a pull-down menu called “purse” that would contain credit cards and other identity items required for doing business in the world. Neither would be owned or provided by one vendor (not even the browser's). Instead they would be features of the user's own identity as an autonomous customer in the marketplace.

In other words, she wanted the browser to be an instrument of demand, rather than of supply. And she had reason to be encouraged. That's what early versions of Netscape seemed to be about.

But our family's view of the browser business was a little biased. One of the top executives at Netscape was a close relative. So we rooted for Netscape in its famous battles with Microsoft. We proudly wore Netscape T-shirts and other schwag. We were also quietly frustrated when the company began to add all kinds of extraneous crap to the browser and to become obsessed with its portal: a site cluttered with partner links through which users would be funneled to troughs of content, some of which were called channels in the manner of television.

Rather than develop ways to empower customers to do business in freshly networked markets, Netscape desperately went looking for ways to “aggregate eyeballs” and make money the old-fashioned way: as a media company. That seems to have made Netscape attractive to AOL but not to its browser users.

Back in 1999, at the height of the dot-com boom, we wrote “markets are getting smarter—and getting smarter faster than most companies” in *The Cluetrain Manifesto*. That thinking was influenced (in my case, at least) by what I saw happening in the development bazaar where Linux was growing and flourishing. Linux was proof that markets were more than just labels that suppliers tacked on demographics and product categories. Like the Net on which it grew, Linux was good for both the demand and the supply side, even when it was just being put to use as available free material rather than as something bought and sold. A market that used Linux resourcefully was more than a smart market; it was ahead of the curve.

What makes markets smart is the ease with which people can share information and expertise. But what is it that makes markets powerful? Is it just what economists call buying power—a sum of money ready to be spent? Would it not also be about how goods get valued and money gets spent?

Most of us are only familiar with a few ways to spend money: cash, checks and credit cards. But what if there were more ways to spend money? And better ways too? Let's take it a step farther. What if markets were comprised of something more than the presence of buyers and sellers transacting business? What if markets were also comprised of relationships? I'm not just talking about the relationships each of us have with our phone companies, our banks, our credit-card companies and various retailers (although those are on the table here). I'm talking about relationships with experts whose opinions we value and whom we might be willing to compensate for informing our buying and selling decisions—if the means were in place and the costs were minimal. One example would be magazines; we could pay a small sum on an à la carte basis for expert opinions they've aggregated about wines, cars, travel, music or other topics.

What if each of us had the power to control the terms of our relationships with every company whose goods and services we value, including the ability to exchange micropayments for microservices and microproducts on an as-needed basis?

What if every channel carried by every cable and satellite TV system were available on an à la carte basis? Yes, you could subscribe, or you could pay as you go—up to you.

The main reason we have a hard time imagining these scenarios isn't just that the technical problems are daunting. It's that we're too accustomed to operating as consumers. We see all our power located in near-binary choices. Which will it be...local cable, DishTV, DirecTV or nothing? End of menu.

Yes, the Net might make us smarter customers than we used to be; but in way too many categories, supply is still in charge.

But what if we stop looking like wallets with eyeballs? What if we start looking like a business interface, with our own native protocols and APIs, wrapped around a unique identity that nobody outside ourselves is free to manage? What if we possess a lot of ways to do business other than presenting cash or credit cards? What if the nature of our personal protocols and APIs allowed continuous relationships within which many transactions might take place over a long period of time—and involve more money, and more value, as both parties get to know and trust each other over time?

This is the kind of stuff that's been going through my mind—and through e-mail, phone and meat-space conversations—ever since Andre Durand told me about PingID and asked me to serve on its advisory board. Andre is the founder of Jabber, Inc., on whose advisory board I also serve. Like Jabber, PingID has a .org and a .com side, with an open-source development model for one and an independent commercial development model for the other. Unlike Jabber, the idea is to create something for everybody, namely an identity framework controlled by the individual rather than by any one company, including PingID. This framework would embody the ability to be anonymous, pseudonymous and polynomolous. Most importantly, it would serve as the infrastructural framework for all kinds of businesses that welcome and value full-power customers, ready not just to talk and to transact but to relate over a period of time.

Andre describes identity as a three-tier affair. At the center is tier one: that's your core identity alone. You're in charge of it. Outside that is tier two. This is the identity issued to you by the government, by retailers, by airlines, by insurance companies, by credit-card companies. Every piece of plastic in your wallet is a tier-two identity. Tier three is the cloud of highly presumptuous identity information held by direct marketers and others who hope you may be the one consumer in 50 who responds to a promotional message.

When tier one becomes fully empowered, tier two suddenly takes on fresh meaning and tier three becomes obsolete. Tier two identities become links over which relationships can grow in both directions. It's up to both parties. More significantly, new relationships, and new kinds of relationships, are invited in.

Let's say your palm phone knows by its GPS that you're in Austin, Texas. You're looking for a good barbecue place. There must be a lot of them, and maybe you could look around on the Web for one, but you'd rather take advantage of local expertise on the matter. Thanks to your relationship with Zagats, information about barbecue places is pushed down to your device by an XML stream. By

prior agreement about these kinds of events, a micropayment is made to Zagats for the information. And since Zagats has similar relationships with its own sources of expert information, some of your micropayment gets passed along to its original sources. Yes, it's easy to imagine all the things that could go wrong here. But what's right is that your relationships are between equally empowered participants. You're not just an infodump at the end of some intermediary's conveyor belt for content of its own choosing.

Andre's ideas here are still new and barely tested. As I write this, development is only beginning on the project. But what has me excited is that for the first time I'm seeing an identity project that respects the individual as both an autonomous and private soul, and as a fully potentiated participant in real markets where customers, vendors and everybody else can operate on agreeable terms, and everybody is fully capable of forming any relationships they want.

If this goes the way I hope and expect it will, the project should be equally attractive to both open-source and independent commercial developers (which are not mutually exclusive categories). And I believe the degree to which it can succeed depends utterly on how well we can jettison the industrial concepts that have enslaved our minds ever since industry won the Industrial Revolution.

Linux gave us a way to do exactly that with operating systems and by helping us create the worldwide commons we call the Net. Apache helped us do the same with web service. Sendmail did it with e-mail. Jabber is starting to do it with instant messaging and XML routing. Maybe PingID can do it with identity. If so, it will contribute to the business ecosystem we call the Net's commons.

This commons is an ideal platform for real markets. Without completely autonomous customers, our markets will stay captive to Supply. Even if they live on the Net.

Resources

Doc Searls is senior editor of *Linux Journal*.

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

[Advanced search](#)

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

Advanced search

Hewlett-Packard x4000 Workstation

Thad J. Beier

Issue #98, June 2002

We tested this x4000 as an artist workstation as well as a batch renderer.

Hewlett-Packard's workstation line includes the x4000, which has a number of configuration options. For our testing, HP delivered a top-of-the-line machine with two 2.2GHz Xeon processors, 4GB of RDRAM memory and a FireGL4 graphics card, running Red Hat 7.1.

Goals of the Review

Our company is a visual effects production company. From our inception we had been using SGI workstations for both interactive desktop use and for batch rendering. About two years ago we began using generic Linux workstations for batch rendering, but we were still using SGI machines for artist workstations. We now feel that the maturity of both Linux and the newer hardware solutions are to a point that they can provide far better artist workstations at a reasonable cost.

We tested this x4000 as an artist workstation, as well as a batch renderer. We installed a wide range of software and performed a representative set of tasks over the course of three weeks.

Hardware

As described above, the configuration is as follows:

- two 2.2GHz Xeon processors, with 512KB cache each
- four 1GB RDRAM
- FireGL4 graphics card
- CD-ROM
- 3¼-inch floppy

- Lucent Winmodem
- Intel Camino 2 Chipset (aka 820e)
- two 53c1010 Ultra3 SCSI adapters
- Intel 82557 Fast Ethernet
- Tehama Chipset AGP Bridge
- Seagate ST318406LW disk, about 16GB
- DVI-HD15 adapter cable

Software

Software is as follows:

- Red Hat version 7.1, kernel 2.4.3-12smp
- Hammerhead's in-house graphics software
- Shake by Nothing Real
- Maya by Alias | Wavefront
- RAYZ by Silicon Grail

Experience as a Workstation

Most of the software that we use was developed in-house and has been ported to Linux from the SGI/IRIX environment over the last couple of years. Installing this software on the x4000 was trivial; everything copied over and worked perfectly with no changes. This is a tribute to the maturity of Linux and the expertise of HP and ATI in building compatible systems.

The x4000 was shockingly faster than the machines we had been using (admittedly old SGI O2 computers). While it is unfair to compare the x4000 to SGI machines made three years ago, it is still revealing. The x4000 with the FireGL4 graphics board could display both 2-D and 3-D information on the order of 30-50 times as fast as our old O2s. Changes by a factor of two or four are amazing, but by a factor of 40 is astonishing.

In the visual effects business, it is often true that 2-D performance is as important as 3-D performance. Much of the work involves working directly with hundreds or thousands of high-resolution film frames, doing rotoscoping by tracing the outlines of regions of an image, tracking features from frame to frame and playing back high-resolution images to examine these frames in motion. The x4000 with the FireGL4 had the best performance I have ever seen, displaying more than 250 million pixels per second. At a resolution of 1,600 × 1,200, it could display 130 frames per second. There was sufficient processing power available to do fairly complex color correction on the fly and still get better than real-time performance.

While our test machine had only a 100-baseTX network card (based on the Intel i82557 chipset), it was dramatically faster at reading frames from our file server than our other Linux boxes, often approaching 10MB/sec over NFS. This allows the animators to be much more productive, as they are not waiting so long for the next image to load.

Of course, 3-D performance is also vitally important; and it is no surprise that this x4000 gave a stellar performance in this area too. In simple tests viewing independent polygons, the machine would render 2.5 million polygons per second. One can use Alias|Wavefront's Maya to edit 100,000 polygon models with true interactive performance. In general, the 3-D performance was so astonishingly better than the performance of our SGI machines that there was no comparison whatsoever; it was also dramatically faster than the RADEON and GeForce cards on our other Linux machines.

Experience as a Batch-Rendering Station

For the three and a half weeks that we had the machine, it was used almost nonstop as a rendering server. The 2.2GHz processors were, on average, about 1.4 times as fast as the 1.3GHz Athlon processors in the rest of our renderfarm. The huge 4GB of memory allowed us to render far more complex scenes at once than we could with our other smaller computers.

The SMP capabilities of the machine were a pleasant surprise, in that the performance was almost perfectly double that of a single-processor machine. Rendering two frames took almost exactly the same time as rendering one; there was little contention for shared resources. Furthermore, the SCSI disk offloaded the processor, allowing the CPUs to run at full speed even during heavy disk access—something not found in commodity ATA-based systems. An artist could use one processor for background rendering, while doing interactive work with the other processor without much contention.

Problems and Glitches

What I was most concerned with in replacing our tried-and-true SGI workstations with Linux-based desktops were the little glitches that lurk in the dark corners, unexpectedly ruining one's experience. Other UNIX-based workstations don't have nearly the price/performance ratio of PCs, but they are relatively bulletproof. The qualification and testing that SGI or Sun put their machines through guarantees some amount of freedom from problems.

HP did their homework here and built a very solid machine. There were none of the display glitches that we find in consumer-grade machines. The FireGL4 was absolutely problem-free, no speckling pixels during refreshes, no polygons

where they shouldn't be, no problems of any kind. The OpenGL drivers seem to be every bit as mature as those on other workstations.

We did have minor problems that might be expected with machines of this scale. Maya refused to run at first, claiming to be out of memory (this turned out paradoxically to be a problem of having too much memory). The 4GB in the machine is too large for the 32-bit integer that Maya was using to store the amount of free memory, wrapping around to a negative value. Booting Linux with mem=2048k yielded perfect Maya performance.

There was a bit of a learning curve for us as well with the window manager. Maya depends on using Alt-mouse button chords to move the camera. It took us longer than it should have to realize that GNOME had already intercepted these events—once we disabled those combinations our camera moves worked as they should.

With RAYZ, there was a minor marking-menu problem, but fairly simple workarounds were suggested by Silicon Grail.

The x4000 has an array of four diagnostic LEDs on the front panel. These started flashing at one point, and the code was deciphered as indicating a low CMOS battery. Reseating the battery, as suggested by the manual, caused the problem to go away.

Conclusion

In general, I found the problems to be remarkably few and easy to deal with. The software packages came up cleanly and easily with no configuration headaches. Companies like HP, Alias|Wavefront, Nothing Real and Silicon Grail recognize the size and importance of the Linux desktop workstation community and devote the required resources to release tested, well-qualified solutions.

HP has made a formidable computer in the x4000. At least in the fully outfitted configuration that we tested, it was a serious competitor to anything from the traditional workstation vendors. All of the advanced commercial software that we loaded worked as expected, as did our in-house software. In every particular area the performance was as high as anything we've seen, and all aspects of the machine worked synergistically to provide a great workstation.

Product Information/The Good/The Bad



Thad Beier is an owner of Hammerhead Productions, a digital film company based in Los Angeles, California. He can be reached at thad@hammerhead.com.

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

[Advanced search](#)

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

Advanced search

Letters

Various

Issue #98, June 2002

Readers sound off.

I read with great interest your piece on open-source sendmail. While you brought up some great points regarding its technology and deployment, there were some inaccuracies that are noted below. The corrections were made by Sendmail, Inc., the commercial entity that was cofounded by sendmail author Eric Allman in 1998.

Article: 8.11, in fact, was a “features” release: rather than being released to patch security holes, it was released because the sendmail team had added support for TLS encryption and for the SMTP-AUTH extension to SMTP.

Correction: SMTP AUTH was released in 8.10, 8.11 adding support for encryption when using SMTP AUTH.

Article: The first and probably longest task in setting up an SMTP gateway is generating `/etc/sendmail.cf`.

Correction: The path is `/etc/mail/sendmail.cf`.

Article: Below is a mailertable with two different actions:

```
polkatistas.org smtp:internalmail.polkatistas.org
mail.polkatistas.org local:postmaster
```

In addition to delivery rules, sendmail needs to know which e-mail destinations should be considered synonyms of the local (SMTP gateway's) hostname. These are specified in `/etc/mail/local-host-names`, one per line:

```
mail.polkatistas.org
```

Correction: mail.polkatistas.org should be in the mailertable *or* local-host-names, not both.

Article: Note that this won't work for aliases, which has its own utility, newaliases. Run newaliases without any flags to convert your changed /etc/aliases file into a new /etc/aliases.db file automatically.

Correction: The path should be /etc/mail/aliases (and /etc/mail/aliases.db).

—Anna Vrechek

Mick replies: First, yes, the official location of aliases, aliases.db and sendmail.cf has changed from /etc/ to /etc/mail. However, on the Linux distributions I used for my testing in writing this article (SuSE 7.1 and Red Hat 7.0), the Linux-vendor-provided sendmail packages installed these files in /etc/, probably for historical reasons (i.e., many users probably still expect them to be there). My article intentionally reflected the reality of most of my readers, even though this differs slightly from the reality of a sendmail distribution built from the latest source or installed from recent sendmail installation media from Sendmail, Inc. Second, I did indeed confuse my feature-add-dates for Sendmail 8.10 vs. 8.11, proving that reading the same release notes repeatedly late at night sometimes has the opposite of its intended effect of clarifying things. Thanks for setting the record straight. Third, thanks also for clarifying the mailertable vs. local-host-names redundancy. All I have to say in my defense is that 1) the method I described does work, even though it's redundant; and 2) sendmail's documentation (as included in the free version) isn't at all clear about the relationship between these files, so I do sincerely appreciate your feedback on this matter.

Hot Article

I'd like to congratulate Greg Kroah-Hartman on his Kernel Korner article "Hot Plug" [*LJ*, April 2002]. I particularly enjoyed the article since it was written in a way that mere mortals such as myself (C++/Qt is far more my style than C, thanks) could understand, whilst still remaining reasonably technical. It also taught me a lot about hot plug—I didn't know anything about it previously. I'd love to see more of this kind of feature. Thanks for a great magazine!

—Chris Howells

Easier Connection

I just got done reading the April 2002 issue of *Linux Journal* and have some comments on Andrew Trice's article "Connect to Microsoft SQL 2000 with the Perl Sybase Module".

First, the original version of Microsoft SQL Server was built on Sybase technology, thus the reason that Sybase connectors can be utilized.

Second, there is a much easier way to connect to SQL Server 2000. SQL Server 2000 can connect to IIS via a virtual directory (and patch, patch, patch!), then a query can be performed by using an HTTP request such as:

```
http://IISServer/nwind?sql=SELECT+*+  
FROM+Customers+FOR+XML+AUTO&root=root
```

This example connects to the nwind database and does a simple select; the data is then returned as an XML document with root as the top-most element. This could then be parsed with Perl in the standard way.

This method doesn't allow access to some of the handy elements of an ODBC connection, and doesn't allow an easy mechanism for inserts/updates, although they can be performed with templates and OpenXML.

If all you need SQL Server for is running ad hoc queries, this is a simple solution.

—David Scott

Missing Antiques

The cover of your April 2002 issue has the words “Antiquing Your Desktop”. Huh? Wherzat? Admittedly, I only scoured the index and flipped through each page, not yet having had time to read the thing cover to cover (okay, *Playboy* arrived at the same time). But I have had my interest seriously piqued by what appears to be a non-existent article. Please, Oh Great Ones, shed some light!

—Andrew Bell

Andrew, the antiquing your desktop referred to Marcel Gagné's article “Interoperate with Me”, where he “revisits a few familiar desktop environments from the past”. I apologize if our sorry little play on words had you looking for something on furniture restoration. I hope you find Marcel's article equally interesting.

—Editor

Just an Addict

I'd like to say that I found Robert Adam's article “The m4 Macro Package” (in the April 2002 issue of *LJ*) to be very interesting and immediately useful. I read it on

the train ride to work and put m4 to good use that very day! Thanks for keeping me hooked on *LJ* with articles like these.

—Fred Daoud

False Hopes

The article “The CodeWeavers CrossOver Plugin” in the April 2002 issue was very exciting to read. However, the hope instilled by the article was false because of one requirement the article failed to mention: an x86 processor. It was very disappointing to discover that requirement only after checking the web page.

I would have thought the writers and editors of *Linux Journal* would have been aware of the existence of instruction set architectures other than x86. There are actually quite a number of different hardware architectures on which Linux runs very well. For my part, I have chosen to use an architecture superior to x86 out of simple disgust for segment registers, prefix bytes, the A20M pin, a register model based on the 8008, etc.

—Robert M. Riches Jr.

Erratum

Please note the correction to the web address on page 89 of the February 2002 issue, from www.mesa.org to www.mesa3d.org, regarding the Mesa 3-D Graphics Library.

—Dirk

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

[Advanced search](#)

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

Advanced search

UpFront

Various

Issue #98, June 2002

LJ Index, Stop the Presses and more.

We're wondering howmany haikus it will takebefore this is over

Our nominee for Free Software of the Week is www.oblomovka.com/code/haiku/haiku. It's a Python script that lets you find accidental haiku in any text file, as we did in the LDP HOWTOs way back in 2000. One Linux mailing list contained these:

Well, Junkbuster won't let me view it. I'm sure it'sfascinating, though.

I would rather chewmy leg off than use Netscapeweb servers again.

Solaris weeniesand retards: /opt/foo should be/usr/lib/foo dammit.

Orinoco hasby far the best range, and avery nice driver.

Bush's passion forliterary works was sparkedat an early age.

You'll believe us. Youhave no choice.
INTERACTIVECOMPUTER SERVICE.

Nielsen has nevercreated a beautifuldesign in his life.

A section in thecathedral bar areais reserved for us.

Even Richard Stallman is a (haiku) poet and doesn't know it. From the gnu.org site, this haiku about the economic harm of proprietary software:

One person gains one dollar by destroying two dollars' worth of wealth.

Our nominee for Free Software of the Month is “mencal—a menstruation calendar—a program looking like the Linux program cal. The difference is that in mencal some days are colored red”: mencal.kyberdigi.cz/english.html.

—Don Marti

High Definition Linux

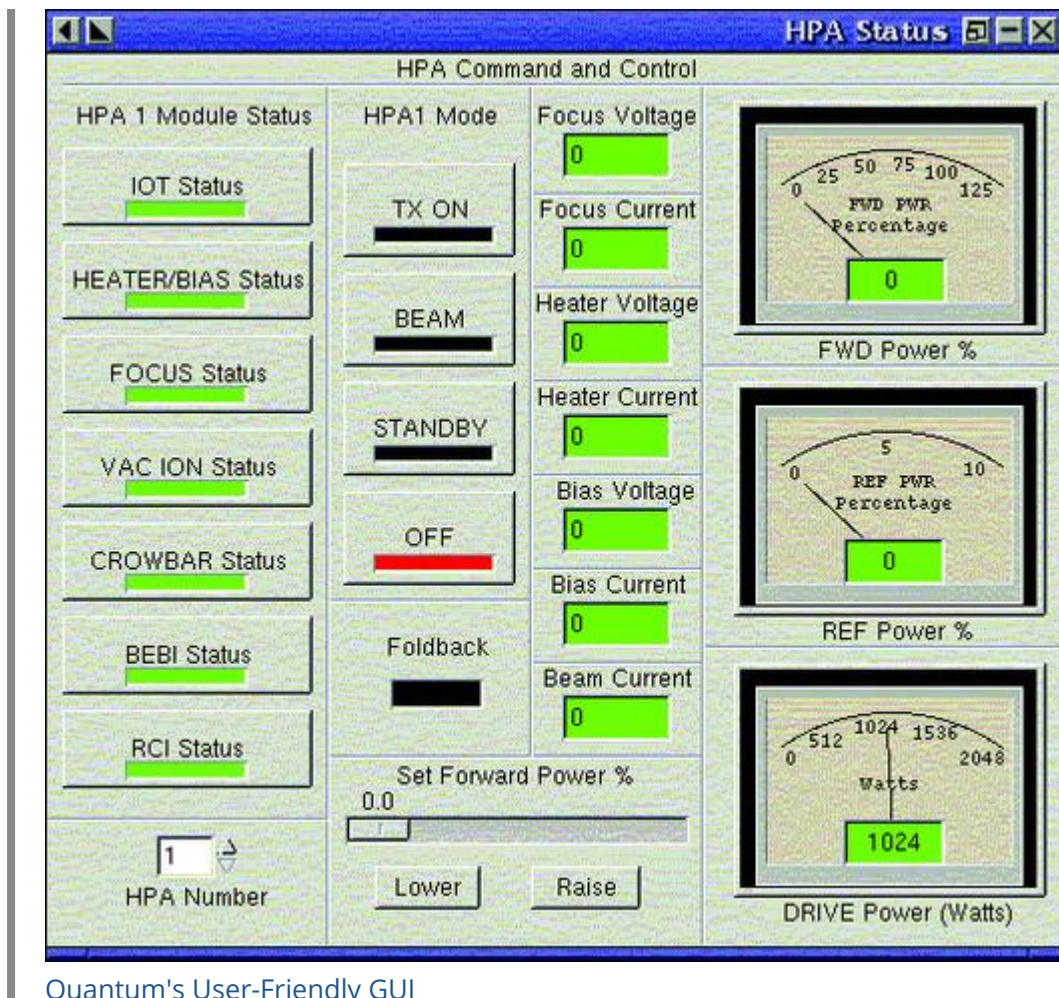
:What's the best looking Linux you've ever seen? Well, if you're among the shrinking number of people who haven't been stunned by HDTV, you may have already seen it. Acrodyne Industries (www.acrodyne.com), the television transmitter division of Sinclair Broadcasting, has a history of industry firsts, and the latest is embedding Linux in their latest transmitters.

Andrew Domonkos, principal systems software engineer with AI, says,

We are the first in our industry to introduce a television transmitter system using Linux as its operating system. Our top-of-the-line HDTV transmitter series, the Quantum, uses an industrial PC running Red Hat Linux in a network server configuration. We chose Linux for both its reliability as well as its networking potential. Running anything else would be unacceptable; when a TV broadcaster goes off the air due to a system controller problem, the loss in revenue can be tremendous.

A paper by Acrodyne's Marc Polovick explains:

Linux (which is internet synonymous) is a UNIX-based OS and provided a valued starting point to this requirement. It was the intent of the Quantum designers to use UNIX as it was designed to be used, in a distributed processing “server/client” environment, a real-time/object-oriented data driven system architecture. A user-friendly GUI is provided and displayed through one large LCD display device. This intuitive graphics-based interface has been designed to be mimicked and provide secure “remote anywhere” monitoring capabilities that are network accessible in almost every network environment.



At the receiving end, Linux has become a common operating system in set-top boxes as well. In fact, National Semiconductor recently created Linux4.TV, "Where open-source Linux meets interactive television"--for engineers working on products that use the company's Geode SC1200 integrated processor and SP1SC10 development platform.

More proof that Linux just keeps looking better.

—Doc Searls

LJ Index—June 2002

1. Percentage of record label audits found to have underpaid artists: 99.99
2. Thousands of cases from which the above numbers were derived: 9
3. Number of cases in the same group in which the artist was overpaid by the label: 1
4. Millions of dollars required to produce a hit record: .5
5. Number of families followed in a 17-year study of TV and violence: 707

6. Range of chance that those watching seven or more hours per week of TV in their formative years would later commit a violent act: 16-116
7. Billions of dollars grossed by Sony in record sales: 4.6
8. Billions of dollars grossed by Sony in electronics sales: 40
9. Percentage of profit margin of the record industry in 1994: 30
10. Range in percentage of profit margin of the record industry in 2002: 8-10
11. Percentage of the American population using the Internet in September 2001: 54
12. Millions of Americans using the Internet in September 2001: 143
13. Percentage of American children aged 5 to 17 using computers at home and school: 90
14. Percentage of American teenagers using the Internet: 75
15. Thousands of dollars Winnebago Industries expected to pay for a Microsoft Corp. Exchange upgrade: 150
16. Thousands of dollars paid by Winnebago for a Linux e-mail system, mostly by upgrading the company IBM mainframe: 26
17. Percentage of mainframe MIPS configured for Linux among all shipped by IBM in Q4 2001: 11
18. Size in square kilometers of the Larsen B ice shelf in Antarctica at the end of 2001: 3,250
19. Weight in billions of tons of the Larsen B ice shelf at the end of 2001: 500
20. Average temperature (Celsius) increase at the Larsen B ice shelf over the past 50 years: 2.5
21. Years over which the Larsen B ice shelf is known to have been stable: 1,800
22. Months it took in 2002 for the Larsen B ice shelf to disintegrate from warming: 2

Sources:

1-3: Simon Renshaw, manager of the Dixie Chicks, in the *New York Times*

4-7: *New York Times*

8-11: US Department of Commerce, Economics and Statistics Administration

12-14: US Department of Commerce

15-17: eWeek

18-22: BBC

Netcraft

Usually the news from the Netcraft Web Server Survey is upbeat for Linux folks, but the latest report (March 2002) isn't. Look at the main graphs at www.netcraft.com/survey, and you'll see a tug of war between Apache and Microsoft IIS web servers that's been going on since 1998. In 2001 IIS made some significant gains on Apache, which held shares upwards of 60%, but those finally seemed to reverse in February 2002. But in March, IIS gained 4.89%, while Apache dropped 4.67%, to percentages among Top Developers of 57.36% for Apache and 34.02% for IIS.

Netcraft says this represents a shift of about two million sites, "primarily as a result of register.com and Network Solutions migrating their domain parking facilities to a Windows front end. Register.com had been serving from Apache on Linux and is in the midst of migrating to Windows", Netcraft reports.

Of course, parked domains are a lot less functional than active ones, so this probably doesn't say much about the real and practical uses of the Web from the server side. But it's still annoying to lose even a few bragging rights share points.

Perhaps when these big registrars finish migrating to the Dark Side, the Apache share erosion will cease, or even turn around.

—Doc Searls

It's Trivial

Questions

Q1. One day in the 1940s, Harvard's famed Mark II, the precursor to today's computers, failed. When the problem was investigated, Grace Hopper and her colleagues found that a moth had lodged itself in the circuits, causing the machine to malfunction. The moth was removed with a pair of tweezers (and later was preserved at the Naval Museum in Dahlgren, along with Homer's logs). What word is derived from this incident?

Q2. We have all heard of various flavours of UNIX: AIX, Digital UNIX, HP-UX, Sun-Solaris, IRIX, SCO UNIX. The question is simple: which organization was responsible for development of Xenix, a version of UNIX that ran on PCs and was compatible with AT&T's System V Definition?

Q3. This person is often known as the Wizard of Woz. When he was 18 years old, he used to sell an illegal device called Blue Box at Homebrew Computers, which enabled people to make free long-distance calls. He used the name Oak

Toebark, and his first call with this device was to the Pope in the Vatican to make a confession. Who is he?

Q4. In the lobby of Cyrix Corporation's headquarters in Texas there is a huge tomb. What famous words are inscribed on the tomb? (Guessable if you think of the competitors.)

Q5. How and where would you most probably read the following:

And the beast shall come forth surrounded by a roiling cloud of vengeance. The house of the unbelievers shall be razed and they shall be scorched to the earth. Their tags shall blink until the end of days.

Q6. The author of this program had worked for a year as a volunteer with a project in southern Uganda. After he returned, he found his heart was still there. So, he made a program that we all probably use: Charityware. According to the author, "You can use and copy it as much as you like, but you are encouraged to make a donation to orphans in Uganda." Who is the author and which program are we talking about?

Q7. Ian Hutton, a Microsoft spokesman, admitted a certain wrongdoing on Microsoft's part. He also complained that Microsoft and its PR firm had been receiving malicious and obscene e-mail from Linux fans. In response to this, Eric S. Raymond said, "One wonders if Hutton believes it would be unprofessional to address strong language to a burglar caught nipping the family silver." What event is being talked about here?

Q8. Charles Augustus Lindbergh was the first *Time*: Man of the Year in 1927. Franklin Delano Roosevelt was *Time*: Man of the Year three times, in 1932, 1934 and 1941. Who was the *Time*: Man of the Year in 1982?

Q9. He was a wild Greek centaur who tried to abduct Deianira, Hercules' new bride, while ferrying her across the River Evenus. Hercules heard her cries and shot an arrow into the heart of this centaur. The teenage author of an open-source security software was looking for a name for his project when he chanced upon this name while flipping through the pages of a mythology book. Immediately taken by how cool it sounded, he named his software after this centaur. Name this centaur/software.

Q10. I am a member of the Merrill Lynch's Technology Advisory Board and appeared as a supporting artist in two record albums: "A Song of Gods Gone Mad" and "Full Circle". I am close to completing a science fiction book titled *Shadows and Stars* and am a Tae Kwon Do Federation 1st Dan Black Belt. Netscape Communications described my paper as a major factor in their

decision to release their client software as open source, and I have never taken any courses in computer science or software engineering. Who am I?

Answers

A1. Apparently, on discovering the moth, Hopper exclaimed, "There is a bug in the computer." From then on, whenever there was a problem with the computer, scientists said they were looking for bugs. Even today, any software mistake is called a bug.

A2. Surprising as it may sound, Xenix was developed by Microsoft Corporation.

A3. Steve Wozniak of Apple Computers fame. According to John T. Draper, the call was made at 4 **A.M.** and went like this: "Is the Pope there? I'm calling from California, and I need to confess!"

A4. Intel Inside

A5. When you type "about:mozilla" in the location bar of Netscape.

A6. Bram Moolenaar and Vim.

A7. The Minecraft test fiasco and its aftermath.

A8. The personal computer. More information is available at www.time.com/time/special/moy/1982.html.

A9. Nessus. Renaud Deraison, the author, was just 18 years old when he wrote it.

A10. Eric S. Raymond

—Sumit Dhar

Stop the Presses: Microsoft, Unisys Partner to Put Up an

Anti-UNIX Web Site that Runs on UNIX

In late March 2002, Microsoft and Unisys launched a publicity campaign against Sun, IBM, Hewlett-Packard and every other competitor that runs UNIX—including Linux—on its big systems. The publicity campaign was scheduled to run for 18 months and cost upwards of the \$25 million that Unisys said it was spending alone on the project.

Rather than compete on the merits of Microsoft's and Unisys' competitive product offerings, the company decided to wage the high-tech equivalent of a negative political campaign. Titled "We have the way out", the campaign incredibly accuses UNIX of cornering customers in an expensive box. One ad says:

No wonder Unix makes you feel boxed in. It ties you to an inflexible system. It requires you to pay for expensive experts. It make you struggle daily with a server environment that's more complex than ever.

As if the words in this ad were not sufficiently irony-free, the campaign's web site, wehavethewayout.com, was found to have been served by Apache on FreeBSD. After that bad publicity became news, the site was moved to Windows, first at the same Verio hosting service, and then at Unisys. But in the process it went dark, displaying either a blank white page or a 403 error message.

Later the page came back up, offering membership in "our ecommunity" and five PDF papers to download. The web page was copyrighted by Unisys Corporation.

The campaign was reportedly intended to promote Unisys' ES7000 server and the Datacenter version of Windows 2000, which can run on machines with as many as 32 processors. Unisys faces an uphill battle in this category—not only with Sun and HP, which have had strong UNIX offerings for many years, but from IBM's new eServers, which run either Linux or Microsoft OSes on systems with up to 16 processors and sell at lower prices than Unisys' ES7000s. CNET also reports that only "a few hundred" ES7000 servers have been sold and that "sales partnerships with Dell, Compaq and Hewlett-Packard have all fallen apart".

Naturally a new site called wehavethewayin.com appeared shortly after wehavethewayout.com fell on its face. With pointers to all the leading UNIX variants, wehavethewayin.com says:

These days, with enough money, a report or statistic can be printed that says anything. Usually this is done because those with enough money want to make others believe their lies. Make sure that you are reading a genuine article and not a paid-for lie". The page is copyrighted by Jon Fields of LinuxFreak.org. Shortly after it went up, Fields reported more than 400,000 hits to the page.

—Doc Searls

They Said It

Ninety percent of everything is crap.

—sturgeon

The TiVo is a wrapper around a bad interface.

—Bruce Sterling

The WWW is bloatware. Finding things is impossible because there's so much stuff out there. Think how much hard drive space is wasted on all kinds of web pages that only .0000000001% of the world ever reads. Since the vast majority of people only go to Yahoo, Ebay and MSN, wouldn't the WWW be better if it only had Yahoo, Ebay and MSN? It would be much more "optimized".

—Joel Spolsky

Let me be clear: Microsoft expects 802.11 and its supersets to be present in most places that people spend time. In corporate offices it will be pervasive. In campuses, hotels, convention centers, airports, shopping centers—virtually everywhere—this 11 megabit and up capability will be there.

—Bill Gates

Yes, Microsoft twisted the market's Invisible Hand until all it could do was salute smartly in the direction of Redmond. But the market is still capable of containing Microsoft's hegemony. Just barely. Maybe. I am far more worried about the entertainment-legislative complex. Because the market has emphatically rejected its business model, it's perilously close to rewriting the software and hardware rules to force the market to comply. The government is both venal and stupid enough to do it.

—David Weinberger

Most people who tell you about the patent system have a stake in it, and so they want you to like it. But patents are like the lottery because they only rarely bring benefits to people. Lotteries invite you to think about winning, never about losing, and it is the same with the patenting system.

—Richard M. Stallman

A gored ox is a vigilant ox.

—Jim Barksdale

Software is like sex. You certainly can pay to watch actors, but some things are more fun to do for yourself.

—Miguel de Icaza

I'm wary of the trend toward things which can undermine the reliability and accuracy of Google.

—Seth David Schoen

I only hope no one tries to call people who copy digital media “software terrorists”.

—Derek K. Miller

It was naïve to assume the Net would stay open. Nothing ever does. Things have not been going well in that very large corporate interests, protecting business models, are leveraging lobbying to pass legislation that is harmful and stupid.

—Mitch Kapor

The Hollings bill is a kind of a setback. It threatens what people on both sides have been trying to achieve....People in the industry are not looking for the lock and key scenarios that the Hollings discussions raise.

—Hillary Rosen

The content industry is now Disney and Fox coming to Congress and saying “you choose...you come up with a solution”. These guys think this is “open source”.

—US Sen. Maria Cantwell (D-Wash)

We also need the buggywhip protection act, mandating that no moving vehicle sold in the US not be equipped with a buggywhip. And the Railway Protection Act, mandating that no mass transportation ticket be sold for a service that moves faster or less expensively than a train ride.

—The Rev. A. K. M. Adam

Imagine if radio came before telephony. People would be saying “Who wants to talk personally? Broadcasting is everything!”

—Bob Frankston

Unlicensed spectrum is the best thing we've ever done.

—FCC Chairman Michael Powell

No man is entitled to the blessings of freedom unless he be vigilant in its preservation.

—Douglas MacArthur

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

[Advanced search](#)

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

[Advanced search](#)

From the Editor

Richard Vernon

Issue #98, June 2002

Now's the time to let that fanaticism pay off by coughing up the evidence.

We recently ran a short humor piece on the *Linux Journal* web site that asked the question "Are you a Linux waif?" (</article/5868>). The article was submitted by a woman who has been married to a "Linux man" for 13 years. She said she'd understand if we didn't want to run it because it might be funny only to her and her husband.

Whether people found it funny I don't know, but for a lot of people it struck close to home and turned out to be one of the most commented articles on the site, with people writing in to share their own Linux waif stories. The level of raving fanaticism, er, I mean enthusiastic devotion, ignited a spark of inspiration in our free-beer-numbed minds. We came up with an idea that will provide the staff of *Linux Journal* with some free entertainment, while giving our readers an opportunity to win a truly fabulous prize.



Marc Merlin, Linux Enthusiast Extraordinaire

In the spirit of the old MTV show *Fanatic*, we will be hosting a contest to discover the most dedicated (rabid) Linux devotee (fanatic). If you can prove to us that you are the superlative Linux zealot, you will win complimentary

passage on the Linux Lunacy 2002 Caribbean cruise and a chance to meet some of the most renowned people in the Linux and Open Source communities, including Linus Torvalds, Ted Ts'o and Guido van Rossum. This year's ports of call include Jamaica, the Cayman Islands and Holland-America's private Caribbean island, Half Moon Cay.

To enter, simply submit some kind of physical evidence of your Linux ardor to

Linux Journal Fanatic Contest PO Box 55549 Seattle, WA 98155-0549

Submissions must be received by July 5, 2002. The grand prize winner and two runners-up will be announced in the September issue of *Linux Journal*. The runners-up will also receive a most excellent prize.

Example forms of submission are essay, film, photographs or any combination thereof. For instance, a short film of you wearing your penguin pajamas to a five-star restaurant while trying to show the waiter how to hack the restaurant's point-of-sale system (running embedded Linux) to keep a more "accurate" account of tips would probably make you a major contender.

We look forward to receiving your entries. Good luck!

Richard Vernon is editor in chief of *Linux Journal*.

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

[Advanced search](#)

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

[Advanced search](#)

Best of Tech Support

Various

Issue #98, June 2002

Our experts answer your technical questions.

I have a 3Com 3C905B-TX network card installed on my server. Unfortunately, the NIC operates only at 10Mbps with either a Lantech FE-1601 or a DLink DES-1005D switch. Can someone help me to turn this NIC into full duplex 100Mbps?

—David Desscan, davidcyberletters@rocketmail.com

Try mii-diag from www.scyld.com/diag/index.html and see what your link partner is advertising. If you don't see 100BaseTx-FD in there, your switches are not configured correctly or the autonegotiation is failing for whatever reason. Your next bet is to force 100Mbit and FD as indicated at www.scyld.com/network/vortex.html. You can try to unload the module and reload it with

```
insmod 3c59x debug=1 options=4 full_duplex=1
```

—Marc Merlin, marc_bts@valinux.com

Tunneling in to the Intranet

Is there a way to set up an SSH connection so that HTTP content from my company's internal site goes through the tunnel to a browser running on my home machine?

—Michael Kaneshige, mjkanes@comcast.net

For one web site, you can redirect a local port on your machine to the web site on the other side of the SSH connection:

```
ssh -L1234:intranetserver:80
```

Then point your browser to <http://localhost:1234/>. Note that some of this may be against your company's security policy. Check with them first.

—Marc Merlin, marc_bts@valinux.com

If you want to browse a lot of internal sites from the outside, you can set up an Apache server with `mod_proxy` on your work system, run the SSH tunnel as above and set your web browser's proxy preference to localhost, port 1234. Apache `mod_proxy` can be found at http://httpd.apache.org/docs/mod/mod_proxy.html. If your company's security people give you crap about this, tell them we said it's okay.

—Don Marti, info@linuxjournal.com

Dead Media Project, Banyan Department

We have several HS-8/112 tapes from an old Banyan network we had. They were recorded on an Exabyte 8200. I am trying to retrieve some data off these tapes via a Linux box and a SCSI Exabyte 8505 tape drive. `tar` told me that it was not a tar archive. Do you know of any free way of retrieving the Banyan data off this tape via Linux?

—Craig Johnson, cjohnson@eypae.com

Yours is not a trivial case. A few pointers for reading other system's tapes on Linux are available. They may not apply directly to your case, but they can provide you with good ideas and pointers. For reading tapes from other UNIX systems on your Linux box, see meteora.ucsd.edu/~pierce/linux_tape.html. Another old but worthwhile HOWTO for getting ideas can be found at www.linux.org/docs/ldp/howto/Ftape-HOWTO.html. The first thing you need to find out is the format the data was saved into the tape. From the hardware standpoint, finding out if there was some data compression mechanism while writing data would help. From the operating system and software standpoint, it could be a UNIX typical format like a tar, a cpio or a dd dump; on the other hand, it could be a proprietary format that was used to dump the data to tape. Also, is it a multivolume backup (more than one tape for all the data) or not? Assuming one of these UNIX-type commands was used for data backup, it is common to find some differences in the default behavior of the command used on the writing machine to the one used on the reading machine. For instance, on Linux, tar giving `-o` as one of the options causes the command to try using older (V7) data formats when writing.

—Felipe Barousse Boué, fbarousse@piensa.com

Stop the Multicast Madness

I have been running a Linux-based server at a high school for about five years. Twice in the last three months our network tech has traced multicast packets that appear to be originating from this server. They are flooding the LAN and essentially bringing legitimate traffic to a stop.

I am running the latest SuSE distribution. I can find nothing that could be causing this problem. I have been using tcpdump for the past three weeks to monitor the server in a standalone LAN environment, but I see no multicast traffic. I also have monitored /proc/net/dev and /proc/net/dev_mcast, and see no multicast traffic.

Is there anything else I can do? The network tech will not let me put the server back on-line until I can assure them the problem will not arise again. Is there a process that runs intermittently that could cause this problem?

—George Moreno, ccgmoreno455@netscape.net

It's hard to say what's causing this traffic, but you may be able to persuade your admin that it will not happen again by filtering outgoing multicast traffic. This can be done with iptables:

```
/sbin/iptables -I OUTPUT -d 224.0.0.0/4 -j DROP
```

You'll still need to determine the source of this problem, but this will at least keep these packets off your network.

—Robert Connoy, rconnoy@penguincomputing.com

If your admin says that your machine is to blame, did he capture what type of traffic he is seeing? This will help find the culprit, whether it's your machine or someone else's.

—Christopher Wingert, cwingert@qualcomm.com

That's Going on Your Permanent Record!

I work for a school system that is in need of a stable database system that can also allow the following: 1) have an admin user work remotely via the Web (through a web browser) and 2) allow the teachers (users) to log in (via the Web through a browser) and enter information into the database(s).

—Maurice Pelletier, mpelletier@sad39.k12.me.us

I suggest you take a look at Zope (www.zope.org). Zope is a web-based content management framework that is extremely scalable and works well. Of course, your applications have to be migrated or probably rewritten on Zope.

—Felipe Barousse Boué, fbarousse@piensa.com

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

[Advanced search](#)

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

[Advanced search](#)

New Products

Heather Mead

Issue #98, June 2002

Lifix Go! 2.0, GVS 9000 2U Rack System, QuickCRC and more.

Lifix Go! 2.0 is a set of software components that provide mobility within a single on-going session between all IP networks, including 802.11a/b/g, Ethernet and cellular networks, such as GPRS and CDMA 2000. Lifix Go! consists of a mobility server, including foreign agent and home agent, and mobile client software. Full support for the RFC3220 Mobile IP standard is offered, as is support for multiple network interfaces on all platforms and support for various VPN clients. RADIUS authentication, authorization and accounting are available in all software components; diameter authentication, authorization and accounting support is available for the mobile client.

Contact Lifix Systems Oy, Yliopistonkatu 5, 3rd Floor, FIN-00100 Helsinki, Finland, info@lifix.fi, www.lifix.fi.

GVS 9000 2U Rack System

Terra Soft Solutions introduced a dual GHz G4 cluster node/server/workstation in a GVS 9000 2U Rack System with Yellow Dog Linux 2.0 and Mac OS X pre-installed. Components of the system include dual 1GHz PowerPC G4 with AltiVec CPUs; a 1GB PC-133 SDRAM and two DIMMs for memory; an 80GB IDE drive for storage; Gigabit, FireWire and USB communications; an ATI RADEON 7500 dual video card; a single 64/66 PCI slot; and an Apple USB keyboard and mouse. The system is designed to be comparable to current Macintosh G4 towers.

Contact Terra Soft Solutions, Inc., 117 West Second Street, Loveland, Colorado 80537, 970-278-9243, presales@terrasoftsolutions.com, www.terrasoftsolutions.com.

QuickCRC

QuickCRC for Linux, an object-oriented design tool that automates CRC (class, responsibilities and collaboration) cards, is now available from Excel Software. With QuickCRC, software designers can identify object classes, relationships and related information before writing code. It uses a diagram workspace for creating card and scenario objects; projects are saved as XML files. QuickCRC can also be used to create and add attributes to cards that are suited for lightweight development approaches or for being front ends to larger modeling efforts using UML. QuickCRC also provides active simulation of an evolving design.

Contact Excel Software, 19 Misty Mesa Court, Placitas, New Mexico 87043, 505-771-3719, info@excelsoftware.com, www.excelsoftware.com.

Token Ring Drivers

Madge Networks has announced the availability of open-source drivers for their range of Token Ring adapters. The new drivers include software that supports version 2.4.2 of the kernel, as well as precompiled drivers for implementations based on the 2.4.2 kernel. Supported by these drivers are Madge's Smart MK4 PCI family, RapidFire 3140 PCI family and the Smart CardBus MK2 Token Ring adapters. The drivers can be downloaded from the Madge Networks site at www.madge.com/software. The site also offers an on-line discussion group for Linux users.

Contact Madge Networks, 1 State Street Plaza, 12th Floor, New York, New York 10004, 800-US-MADGE (toll-free), www.madge.com.

EasiLiX SM

EasiLiX SM is software for network security and administration management. Designed to be installed and maintained easily and quickly, EasiLiX includes tools for DNS, Web, e-mail, Squid proxy, FTP and DHCP services for intranet, internet and network needs. EasiLiX shares a single IP address across all computers on the network, and the network can be configured with the included DNS and DHCP servers. Other features of EasiLiX include intergration with kernel version 2.4; ReiserFS; Ethernet, xDSL, dial-up and cable modem support; 128bit SSL and OpenSSL support; Samba file/printer sharing with other desktop systems; MySQL database; and PHP, Perl and Python scripting support.

Contact Easilize Software Solutions, info@easilize.com, www.easilize.com.

3DBOXX R1 and RenderBOXX R Series

BOXX Technologies, providers of digital content creation systems, introduced the 3DBOXX R1 series of workstations and the RenderBOXX R series of rendering systems. Both systems feature the AMD Athlon MP processor 2000+ with QuantiSpeed architecture and Smart MP technology. They are available in dual-processor configurations, feature NVIDIA Quadro4 XGL graphics adapters and are optimized for creating and rendering 3-D content and animation using popular software such as Maya, Houdini and others. The AMD Athlon MP processor is designed for high-performance multiprocessing servers and workstations.

Contact BOXX Technologies, Inc., 9390 Research Boulevard, Kaleido II, Suite 300, Austin, Texas 78759, 877-877-2699 (toll-free), www.boxxtech.com.

Red Hat Linux Advanced Server

Red Hat's enterprise-level platform, Red Hat Linux Advanced Server, offers products that enable large companies to accelerate migration to Linux running on Intel-based server systems. It features kernel enhancements for improved application I/O performance (asynchronous I/O) and scalable data access across multiple servers. Clustering benefits include NFS/CIFS failover, scalable node support, multiple load-balancing configurations, 2-node failover and shared-storage devices. Advanced Server also offers an improved process scheduler, support for up to eight SMPs, global 24/7 support and long-term API consistency.

Contact Red Hat, Inc., PO Box 13588, RTP, North Carolina 27709, 919-754-3700, www.redhat.com.

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

[Advanced search](#)

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