Linux Journal Issue #101/September 2002

Features

What Has 1.1 Terabytes, 9,503 BogoMips and Flies? *by Don Marti*
With a collection of hot hardware, Mr. Marti shows that you can't judge a box by its color.

Indepth

Coding between Mouse and Keyboard, Part I *by Patricia Jung*
In the first part of this two-part article, Jung provides a working example of building GUI apps with Qt.

Bring an Atomic Clock to Your Home with Chrony *by Fred Mora*
Be the first on your block to have atomic clock accuracly on your desktop!

CVS homedir *by Joey Hess*
Ever thought of living your life in CVS? Hess shows how.

Linux Multimedia with Pd and GEM: a User's Report *by Dave Phillips*
Phillips reveals how the Pd sound synthesis and processing environment works to make Linux a viable multimedia platform.

Free Software in Brazil *by Jon Hall*
maddog gives the lowdown on some impressive Brazilian free software projects.

2002 Editors' Choice Awards
Nineteen categories and 21 winners—read all about it.

Embedded

Embedded Perspective *by Rick Lehrbaum*

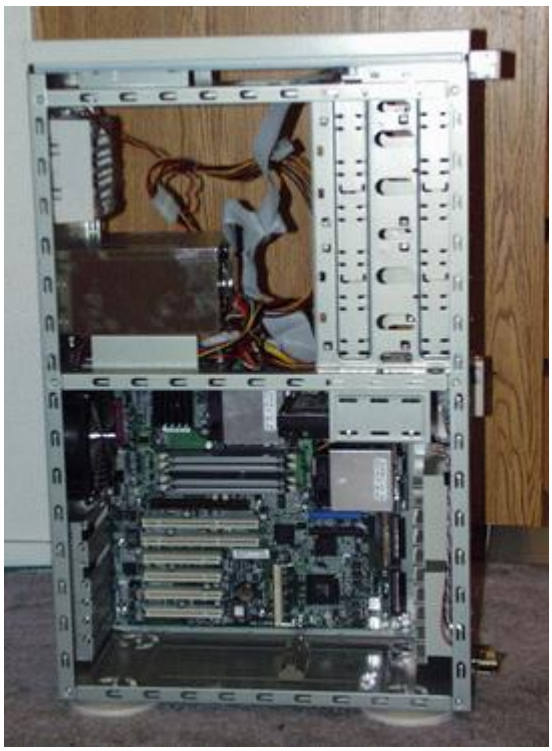
What Has 1.1 Terabytes, 9,503 BogoMips and Flies?

Don Marti

Issue #101, September 2002

What makes the ULB different from a hot PC with Linux on it? We pick stable, well-supported hardware for a homebrew system and end up with a terabyte of storage in an unassuming beige box.

We've been doing “building the Ultimate Linux Box” articles in *LJ* since 1996. In that time, we have seen Linux scale to IBM mainframes, 32-way ccNUMA boxes and other exotic hardware. As much as we'd like to explain how to build them, most of us aren't going to have the space, power or budget for a 32-processor system that we can tear down whenever we want to put in another sound card.



A large tower case has plenty of space for working inside and keeps cables out of the way. You can mount all drives on the upper level, so plenty of air can get to the hot processors. This is before installing drives and cards.

So our Ultimate Linux Box is more of an ultimate Linux workstation or an ultimate Linux small server—big enough to be faster than you need for most uses and small enough to be practical in an office or home environment.

The first question, of course, is whether to buy or build your machine. On the one hand, you can do things for your hand-built system that a mass-market PC manufacturer probably won't:

1. Pick the very best case for the exact hardware you want to run, your work environment, your aesthetic sensibilities and your desire to work inside easily.
2. Use a top-quality power supply and quiet, ball-bearing fans.
3. For non-ultimate systems, you can put a really good SCSI card, SCSI drives and Ethernet card on what is otherwise a low-end desktop machine. This type of configuration will work well for most Linux developer workstations and small server tasks, but mass-market vendors won't usually build them because they don't appeal to people who buy systems by comparing CPU clock speeds and prices.
4. Leave off the parts that will end up gathering dust: your keyboards and mice will last several generations of hardware, and you probably already have a box with a CD burner.

On the other hand, there are two things that a mass-market PC manufacturer can provide that you probably won't:

1. Professional thermal and acoustic engineering. Your homebrew system will likely end up with more fans, a bigger case and a bigger power supply than a mass-market system with similar performance numbers. This doesn't necessarily have to translate into more noise, if you're careful.
2. Relationships with hardware vendors who don't respect you. One of the big differences between our box and the Hewlett-Packard x4000 we reviewed in the *LJ* June 2002 issue is that we're using an ATI video card, and HP uses NVIDIA.

A lot of NVIDIA-based cards are on the market right now, and many users are reporting high performance with NVIDIA's proprietary drivers. If you're Hewlett-Packard and have a close relationship and nondisclosure agreements with NVIDIA, it's possible to make this work. Customers can treat HP like any other UNIX workstation maker—call HP and let them bug NVIDIA if the problem is NVIDIA's fault.

But if you're working with interesting new versions of core software, such as the kernel and X, you might not find much help from the traditional Linux channels for proprietary modules. At USENIX this year, Linus Torvalds said,

“They may work, but you're not getting the full advantage of Linux.” The only case when you could possibly accept a non-Linux kernel module is when you're not doing any “Linux-y” stuff with the box—if you treat it like just another PC and don't recompile the kernel or hack anything whatsoever. Oh, and if you completely trust your hardware vendor.

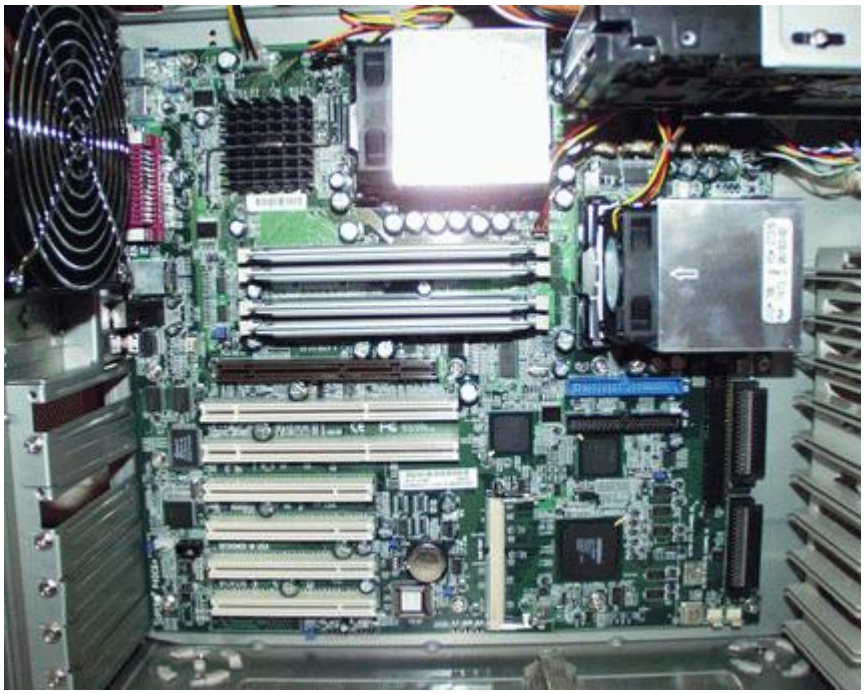
The intermediate route, which is ordering your system from a low-volume or custom shop serving the Linux market, is worth considering if you want to save shopping and building time, as well as get good advice on Linux-friendly parts. Linux hardware vendors are an informal, peer-to-peer reputation system for selecting good parts, and this works surprisingly well. You can go to the web sites of the good ones, see complete lists of every piece of hardware that will go into your system and get a no-hassle warranty on the complete box.

For this year's Ultimate Linux Box, we started with a base configuration of the Glacier Dual Xeon workstation from Aspen Systems Inc. in Colorado, and here's why. Think of the Ultimate Linux Box as a Beowulf node, one with good graphics and sound and a lot of reliable storage, in a tower case.

Companies that build Beowulf clusters are good places to look for the fastest processors and motherboards and the most reliable memory, because cluster users are picky about such things.

Processors, Motherboard and Memory

Unfortunately, our schedule for this article caught us in the middle of Intel's much-awaited transition from RAMBUS to DDR memory. We had to go to press before we could get our hands on a dual Xeon motherboard with both DDR and AGP support, which should be available soon. But if you're playing in Ultimate Linux Box territory, dual Xeon is the way to go. So, we got the RAMBUS-based SuperMicro P4DC6 recommended by Alan Taub at Aspen. Our box clocked out at 9,503 BogoMips with a 2.4.18 kernel. Yow!



SCSI on the motherboard is cheaper than a separate SCSI card. Big fans are good. CPU coolers are shiny.

Since we're writing in the bad old days of RAMBUS for all you happy future people in DDR-land, the best we can offer in the motherboard department is a bunch of mindless platitudes. So do yourself a favor and check the web sites of people who build Linux boxes and then have to take the phone calls when customers have problems with them.

Four features commonly found on some motherboards but not on others are SCSI, Ethernet, sound and video. Don't rule out a motherboard because it has something you won't use. Due to the size of the Linux network server market, all the common Ethernet chipsets you'll find on motherboards, such as the Intel EtherExpress Pro100, are well supported. And if you're planning to build a SCSI system, the price difference between a motherboard with and without SCSI is generally less than the price of a SCSI card.

None of the video or sound chipsets they put on motherboards are Ultimate Linux Box-class, but if you're considering reusing the motherboard for a server later, it doesn't hurt. If you're like many office Linux users and rarely use sound, you might as well use what comes with the motherboard.

Graphics and Sound

The trickiest part of building a Linux system is 3-D graphics. We chose an ATI video card over an NVIDIA one this year. See Frank LaMonica's Sidebar on some possible effects of this choice. Monarch Computer Systems hooked us up with a Hercules 3-D Prophet, which is a nice RADEON 8500-based card that will enable

you to start working with the cutting-edge, open-source clean 3-D drivers when they come out.

Which 3-D Card for Linux?

The sound support front is a happier place. We used the ALSA drivers, and with a properly set up ALSA-based system you shouldn't need to disable the sound chipset on the motherboard to use a high-end sound card. You can use both. We would run the motherboard's audio in and out as a dedicated conference system to chat with headquarters, while saving the Sound Blaster Live! for playing Ogg Vorbis files on headphones. We still like the now-inexpensive Sound Blaster Live! for the ubiquitous kernel support, easy setup, good sound and, most important, the fact that 32 programs can have the audio device open at the same time.

What's All This about a Terabyte?

Until now, we've always recommended a safe, high-performance but expensive choice for storage: two of the fastest-spinning SCSI hard drives you can find. This is still a good conservative option. However, when some people from 3ware showed up at a Silicon Valley Linux Users Group meeting with an Escalade 7850 Storage Switch, we decided to try it out.

So this is the first time we've had RAID 5 and a terabyte of storage on a workstation, and it was surprisingly easy to get working. We had to reboot from an MS-DOS disk to run the utility to update the 7850's firmware, but the current version of the driver is in the 2.4.18 kernel. Red Hat 7.3 and SuSE 8.0, the two distributions we tried, recognized the array out of the box.

You'll be surprised to find the 3ware driver in /usr/src/linux/drivers/scsi, not in drivers/ide. From the kernel's point of view, the ATA RAID controller looks like a SCSI device. You can't make a terabyte filesystem on pre-2.4.18 kernels, so be sure you have 2.4.18 or later.

The web-based management utility, 3DM, that currently ships with the hardware is proprietary, but 3ware assures us that some time in July there will be a scriptable, command-line management tool under the GPL. 3DM presents a clean, easy-to-use interface that is simple to install. You also can rebuild an entire RAID array from a web form.

If you want to use 3DM, you might need to reconfigure your web browser. 3DM runs on port 1080, and Mozilla reported that, “Access to the port number given has been disabled for security reasons.” To override this, add the line:

```
pref("network.security.ports.banned.override",
     "1080");
```

to Mozilla's all.js configuration file, which probably lives in usr/lib/mozilla/defaults/pref/all.js.

3ware claims to be able to do hot-swap if you have the appropriate drive cage. However, we installed the drives normally and didn't test this functionality. We did get excellent performance numbers though. With eight Maxtor ATA drives, we got 173.1MB/s reads and 23.5MB/s writes to an ext3 filesystem on an otherwise unloaded system. This was about the same read performance as a single 10,000RPM SCSI drive, but almost six times as fast for writes.

Driver author Adam Radford recommends two /proc tweaks to speed things up, and we used them. Set /proc/sys/vm/max-readahead to 256, and set /proc/sys/vm/min-readahead to 128.

Case and Building Hints

You don't want to make an Ultimate Linux Box your first PC-building project because of the sheer cost of the parts. Salvage one good box out of a desktop machine with a bad hard drive and a server with a bad motherboard, or something like that. We assume that you know the basics of static protection, reading the Fine Manual for the hardware and not electrocuting yourself. If your Linux box-building tool of choice is a web form, you can skip this part.

Besides the basics, some tools you probably will want are a hemostat for fetching dropped screws and moving jumpers, wire cutters for removing cable ties, a nut driver set, a power screwdriver cranked down to minimum torque and a label maker.

Big tower cases are a win when you're putting together your own box; you have the flexibility to put what you want where you want. We still like the Lian-Li aluminum cases, one of which we used last year.

Our least favorite color is Dumb PC Beige, no matter what you call it. (“Putty” is a typical vendor name for this ugly color. Cool, just what I'd like to have in my den, a big block of putty!) Unfortunately, many of the cool-looking cases in other colors are designed for gamers who run hot processors and video cards but only one or two ATA drives. So it is with a heavy heart that we must recommend the SuperMicro sc760 series, not only for its capacious size but also for its excellent flexibility in drive and fan placement.

SuperMicro cases are available with many convenient spots for drives and fans, many of which you won't have to use. Different models support motherboards with and without the special Xeon mounting holes, so decide on your

motherboard before selecting the case. Working inside is easy: remove the locking front panel and both sides open out like a book.

SuperMicro reduces cooling difficulties with vents on the side of the case.

Think of the case, or any well-designed full tower, as a two-story building—the motherboard and expansion cards live on the bottom floor, and the drives and power supply live upstairs. There is one 12cm exhaust fan directly behind the processors on the first floor, and you can place up to three intake fans at the front. Upstairs, there is one exhaust fan above the power supply, and you can mount up to four fans on the sides of the drives. We recommend leaving off the drive carrier downstairs, if you can, and putting your hard drives in the 5.25" bays with adapter brackets. This gives you more usable intake space at the front of the bottom floor and puts the drives where the side fans can blow on them.

All the ribbon cables from the 3ware card to the drives add up to a surface area of about two and a half square feet. That much cable, placed sloppily, would block a lot of airflow. We bundled the cables together into a flat bunch with Velcro ties and looped the extra length upstairs behind the drives instead of downstairs around the motherboard. Because we have side fans, this is the safest place for it.

It's always important to balance intake and exhaust fans. The natural instinct is to “blow out the hot air”, but too many exhaust fans will drop the pressure

inside the case, canceling out the efforts of the power supply fan and trapping hot air inside the power supply. When the power supply goes, it generally takes something more expensive with it. Because hot air rises, it's hard to go wrong with low front intake fans and high rear exhaust fans.

A nice touch you will probably want to add is thumbscrews in place of the Philips screws for the case sides. The Lian-Li case is already fitted with thumbscrews throughout because the aluminum is very soft. Don't use a screwdriver, manual or power, on an aluminum case.

All the surfaces of the SC760 are paintable, drillable and are easily removable without unplugging a single wire. If you want candy-apple red and a blowhole, you can easily take all the beige stuff on a nice trip to the sheet metal shop and the spray booth without moving drives or cards.

Miscellany

You'll notice that we didn't mention all the peripherals, such as keyboards, mice and monitors. Monitors, keyboards and mice are a matter of personal aesthetic judgment, and your ability to pick one hands-on is probably going to be pretty good. How many computers have your current favorite keyboard and mouse outlasted, anyway?

What about CD burners, DVD drives and tape drives? Well, in a recent reader survey, we saw that 95% of our readers have multiple computers. We recommend that you set up one of your other machines as a backup server or CD ripping/burning station.

That should be enough information to get you started on the journey to custom Linux box building, so in the immortal words of the SuSE installer, “have a lot of fun”.

Don Marti is technical editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Coding between Mouse and Keyboard, Part I

Patricia Jung

Issue #101, September 2002

This article shows you how to create the GUI of a tiny text editor without being a C++ guru. In Part II, we'll add missing functionality and translate the program into languages other than English.

The days when writing GUI applications was a matter between you and your favorite text editor struggling with a compiler and **make** are long gone. Even without an integrated development environment (IDE) like KDevelop, a lot of helper applications promise to make C++ GUI programming as painless as possible.

Whoever expects the KDE-proven Qt toolkit to be a tarball consisting of one library will be disappointed when fetching more than 14MB from ftp.trolltech.com/pub/qt/source. Though Qt was known for its good quality in previous versions, Qt 3.0 had so many bugs that since its first appearance in mid-October 2001, already the fourth maintenance release has come out. It is expected that more will be following. Therefore, it is strongly recommended to choose the latest version.

The qt-x11-free-3.0.x.tar.gz (we used Qt 3.0.4) archive includes not only the class library and documentation in HTML format but also several tools that promise to make life easier for everyone involved in a GUI application project.

Let's pick the new version of Qt Designer to create the graphical interface of a simple text editor application. [The entire code for this editor is available at www.trish.de/pub/linuxjournal/ljeditor_qt3.0.4.] With it comes the user-interface compiler, uic, that converts the Designer's XML output format into C++ code. In Part II of this article we'll actually write some C++ code with your favorite editor and fill the GUI framework with life.

Then, we'll use a new member of the Qt family, Qt Linguist, to localize the program. Like Qt Designer, this is a GUI application using an XML format for

input and output. To create the XML list of phrases requiring translation, it ships with a command-line tool named lupdate. Once they have been translated, another command-line tool, lrelease, converts the XML into the binary format required at application runtime.

Writing Makefiles for Qt applications is not really a trivial thing. Older versions of Qt did not ship with a helper application, but Qt vendor Trolltech offered a tool called tmake for separate download. The Qt 3.0 tarball includes the new qmake utility that comes in handy to create the Makefile for the project. This again we leave for Part II of this article.

Making Plans

With the g++ compiler we have all necessary tools together and should think about the text editor application itself. What should it be able to do?

Obviously we want it to offer the usual tasks: opening a new editor window, opening a file, saving its contents, saving it with a different filename, closing the current editor window and quitting the entire application. Moreover, it should support copy, cut and paste, undo and redo. We want the program to be able to switch font characteristics to italic, bold, underlined and any combination of these. In an About widget, the editor (let's call it ljedit) should reveal some information about itself.

Apart from the changing of font characteristics, all tasks should be available via a File, an Edit or a Help menu. The italics, bold and underline toggling should be done via a submenu in the toolbar populated with additional icons for opening and saving a file, undo, redo, cut, copy and paste.

Each of these icons should open a balloon help when the user hesitates over it with the mouse. Apart from the Save As and the Exit action, all other tasks should be available via keyboard shortcuts.

To avoid users losing data unexpectedly, opening and closing a file as well as exiting the application should be backed by a user dialog that asks whether the old data should be saved or discarded, or whether the user wants to stay with the old file. While most of the above-mentioned tasks can be done with the Designer, this last bit will have to wait until next issue.

Designing the User Interface

If more than one version of Qt is installed on the system, the QTDIR variable should first be set to the directory containing the relevant Qt version. Next, it's time to fire up the Designer with a **designer &** in a shell. In case the directory $QTDIR/bin has not been included in the search path or several Qt versions are

available, the absolute path must be added to the command. To start a new project choose New from the File menu. In the upcoming dialog, click the C++ Project icon and confirm your choice with the OK button. Now we have the opportunity to create a new qmake project file by choosing name, location and adding a project description (Figure 1).

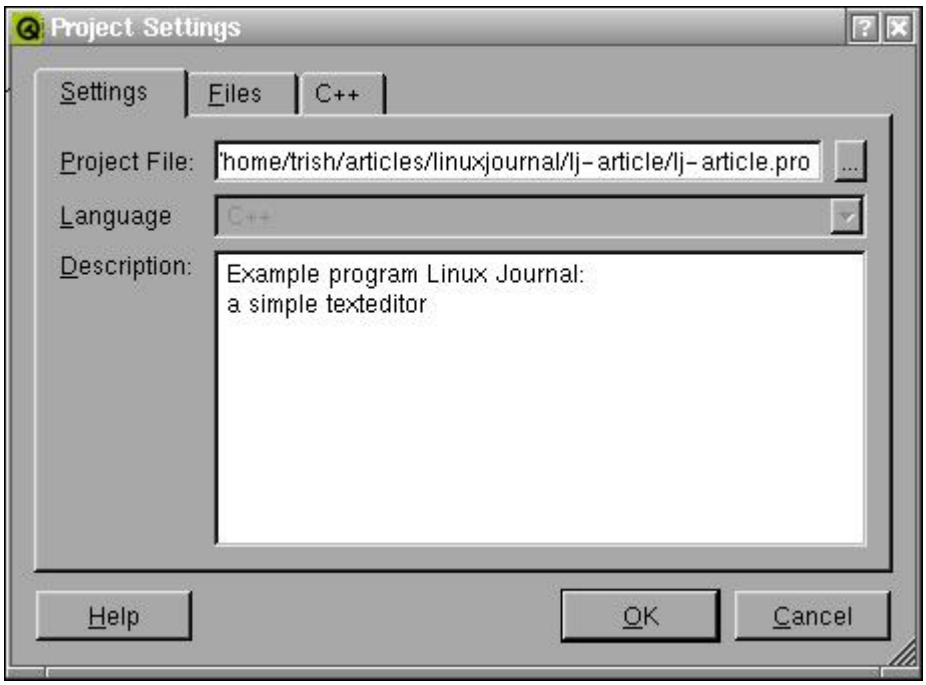


Figure 1. First Step: Creating a Project

Next, we choose the New entry from the File menu once again in order to create the first (and for the purposes of this article only) GUI widget of our editor. Thus, Main Window is the right entry to choose from the New File dialog. We're happy with the default Insert entry that makes it a part of our new project.

This opens the Main Window Wizard. With the first questionnaire (see Figure 2) we feel that yes, Designer should create menus and the toolbar for us. It should provide us with a code framework for the functions used and connect these functions to the relevant actions.

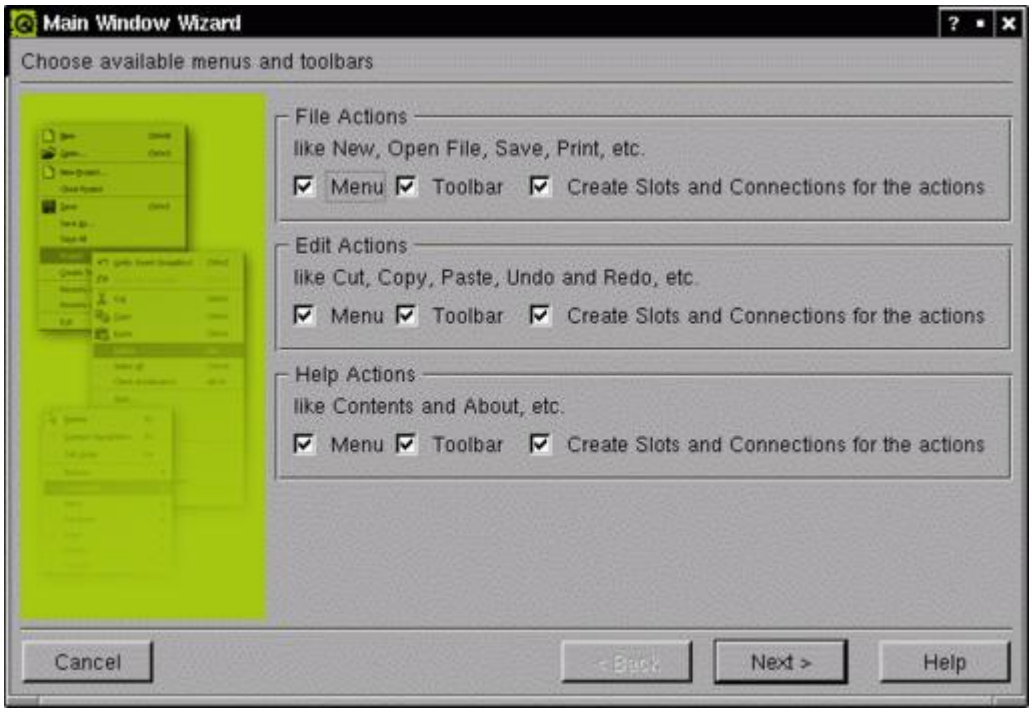


Figure 2. Creates Menus and Toolbar: the Main Window Wizard

The Next> button brings us to a setup dialog for the toolbar. From the File Category, we choose Open and Save and add them to the Toolbar list using the arrow to the right. Then we choose the Edit Category and add the Undo, Redo, Cut, Copy and Paste actions, plus Separators (Figure 3). The last wizard dialog does not leave any work for us to do; it simply finishes the widget form. This leaves us with a Designer looking like Figure 4.

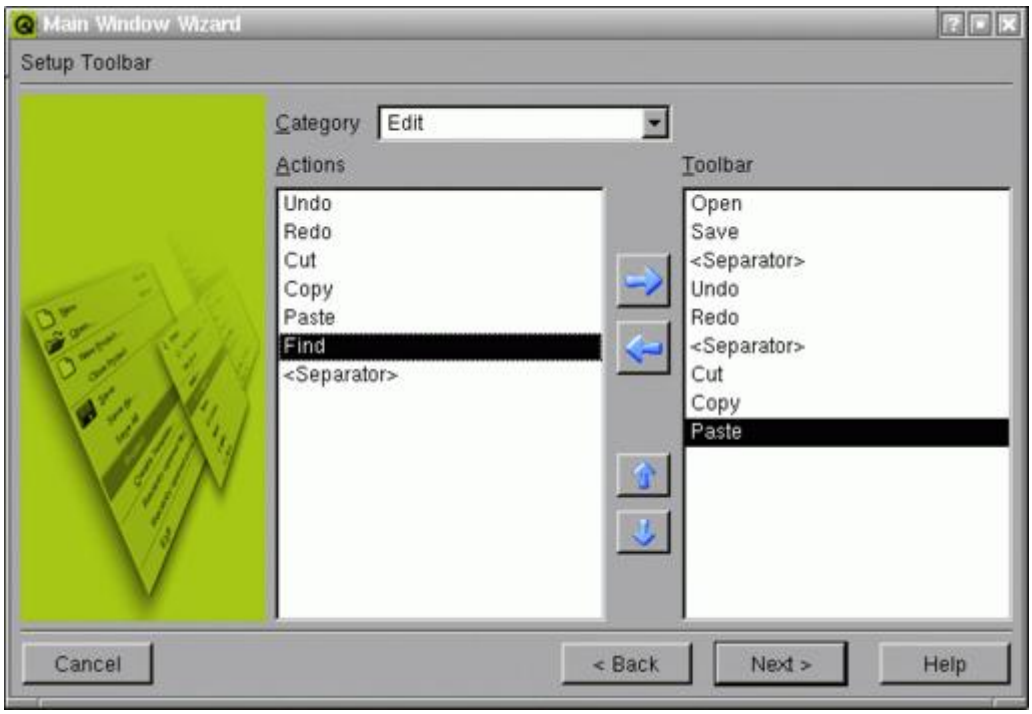


Figure 3. Filling the Toolbar with Actions

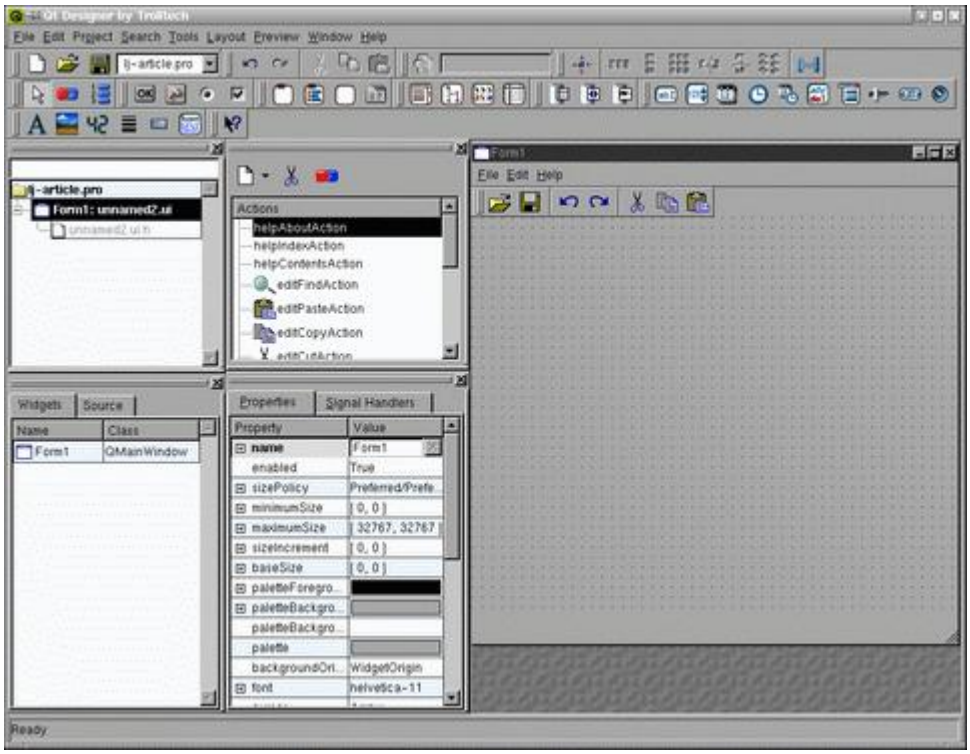


Figure 4. The Designer with a New Main Window

All the new widget is missing in order to look like a proper editor is a nice name in the window caption (currently reading Form1) and the text edit canvas. To solve the first task we need to have a look at the Properties tab in the Property Editor window. It always fills itself with the characteristics of the current widget (i.e., the one last clicked on the form); to begin with this is the form widget.

By changing the property *name* to “ljeditor” we define the class name of the widget we're creating. On the other hand, *caption* defines the window caption and should be set to the new application's name, “ljedit”.

Now, let's add the editor canvas. Click on the Richtext Editor icon (in Figure 4, the fifth icon from the right labeled “abcde”), then click on the rastered background of the ljeditor form, and the new editor canvas can be resized by pulling it in shape with the blue handler points. Let's baptize it “TextEdit” in the Property Editor.

All we have to do now is adjust functionality. First we remove the actions we don't want to implement from the Action Editor (see Figure 5) by clicking on the scissors icon or the Delete Action entry in the context menu available with a right-click onto the marked action. In this example, we can do without helpIndexAction, helpContentsAction, editFindAction and filePrintAction.

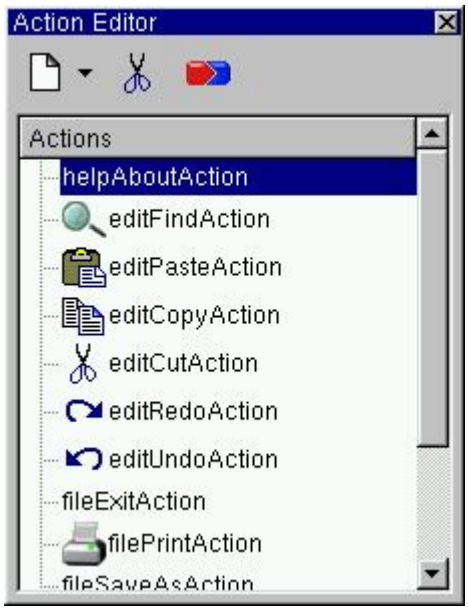


Figure 5. The Action Editor

Action!

In return, an editor needs some new actions to change the font characteristics. In the menu descending from the Action Editor's Create New Action icon (the little paper sheet on the left-hand side) we choose New Dropdown Action Group as a container for them and edit its properties. First, the action group is named “fontCharacter” in the name line of the Property Editor. Then we choose an appropriate icon using the “...” button next to the clicked iconSet property. Using the Add... button, it is possible to add icons stored somewhere in the filesystem.

Its *text*, automatically used for the *menuText* and the various tips (see Glossary), should read “Font Characteristics”. We change the tooltip and the statustip to “Choose font characteristics”, and most importantly, we set the exclusive property to False. This means that the user will be able to combine italics, bold and underlined font if needed. An exclusive action group would allow for only one of them at a time.

Glossary

With a right-click on the fontCharacter action group in the Action Editor and subsequent decision for New Action in the context menu, we add the italics action with Ctrl-I as accelerator key and Italics as text. Because the user can toggle italics on and off, it is important that the property toggleAction reads True. To begin with, italics should be off; therefore, the *on* property must have the value False.

We define the other two child actions of fontCharacter the same way, bold (Ctrl-B) and underline (Ctrl-U).

To add the entire fontCharacter action group to the toolbar, we simply drag it from the Action Editor to the form and drop it onto the toolbar. A right-click into the toolbar allows us to add a separator (Insert Separator).

By now, these actions don't do anything when playing with the preview available by pressing Ctrl-T. To let them actually do what they promise, we once again mark the italics action in the Action Editor, click on the red-blue Edit Connections icon and choose the toggled(bool) signal from the Signals list. Instead of connecting it to an ljeditor slot, we choose TextEdit in the Slots drop-down menu and subsequently setItalic(bool) in the list of slots provided by the QTextEdit class of which TextEdit is a member. No additional click is needed; with the appearance of the connection in the Connections: window, everything is done (see Figure 6), and the OK button is our friend.

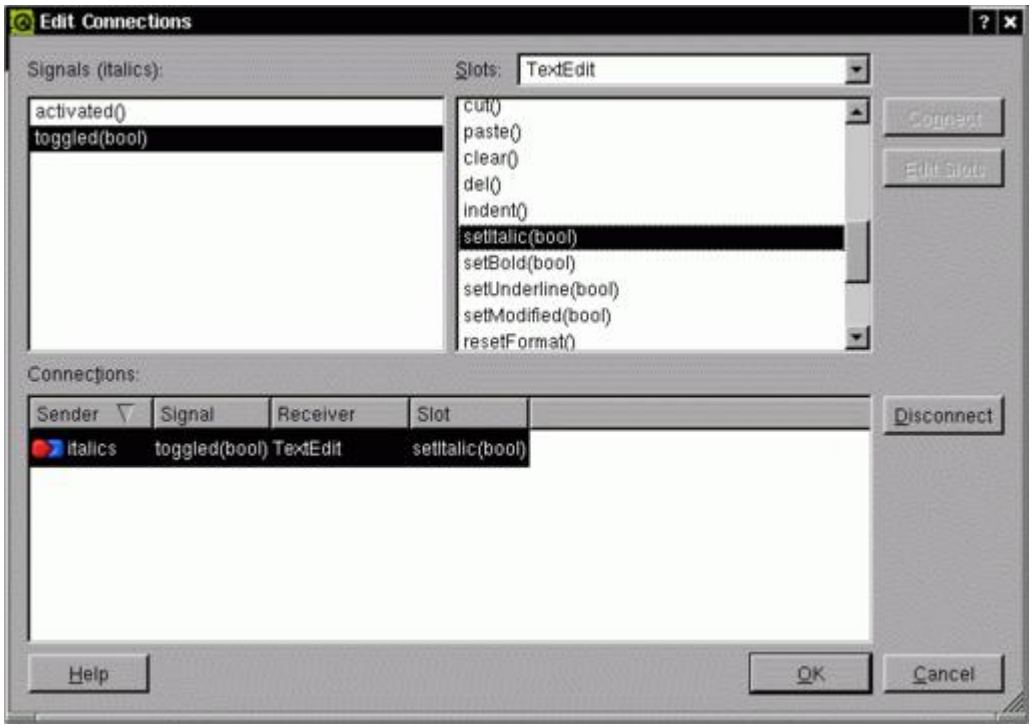


Figure 6. When the italics action is toggled, ljedit writes italics or stops doing so.

Then we repeat the same procedure with the bold action's toggled(bool) signal and TextEdit's setBold(bool) slot. We connect the underline action to TextEdit's setUnderline(bool) slot. After this the preview reacts to Ctrl-I, Ctrl-B and Ctrl-U as wanted.

This encourages us to edit the connections of the predefined actions; these can't be toggled. Instead, they launch a user command like “save current data” when activated (i.e., clicked or chosen). That's why we connect the activated() signal to the appropriate slot.

For editRedoAction this is TextEdit::redo(). We disconnect the connection with the default ljeditor::editRedo(); it would be the right choice if we didn't want to

rely on QTextEdit's redo() function. The same way, editUndoAction's activated() signal is connected with TextEdit::undo() and disconnected from the respective ljeditor function skeleton. We repeat this step with editPasteAction and the TextEdit::paste() slot editCopyAction and the TextEdit::copy() slot, and editCutAction and the TextEdit::cut() slot.

The remaining predefined actions (helpAboutAction, fileExitAction, fileSaveAction, fileSaveAsAction, fileOpenAction and fileNewAction) stay connected with the predefined ljeditor slot skeletons that we later have to fill with code.

One action is still missing: the one that the user activates to close the current editor window (as opposed to fileExitAction, which quits the entire application).

The work flow is a familiar one: we choose New Action from the Action Editor's Create new Action menu. The new action is named “fileCloseAction” in the Property Editor and is equipped with Close as text and Ctrl-Z as the keyboard accelerator.

However, we're missing a slot to connect its activated() signal. We fix this by opening the Slots... menu item in the Designer's Edit menu. Figure 7 shows the dialog that is opened now.

We add a slot with the function name fileClose(), the Return Type void and the Access type public. The Edit Slots dialog does no magic, it simply creates a function skeleton in the class definition of the ljeditor class.

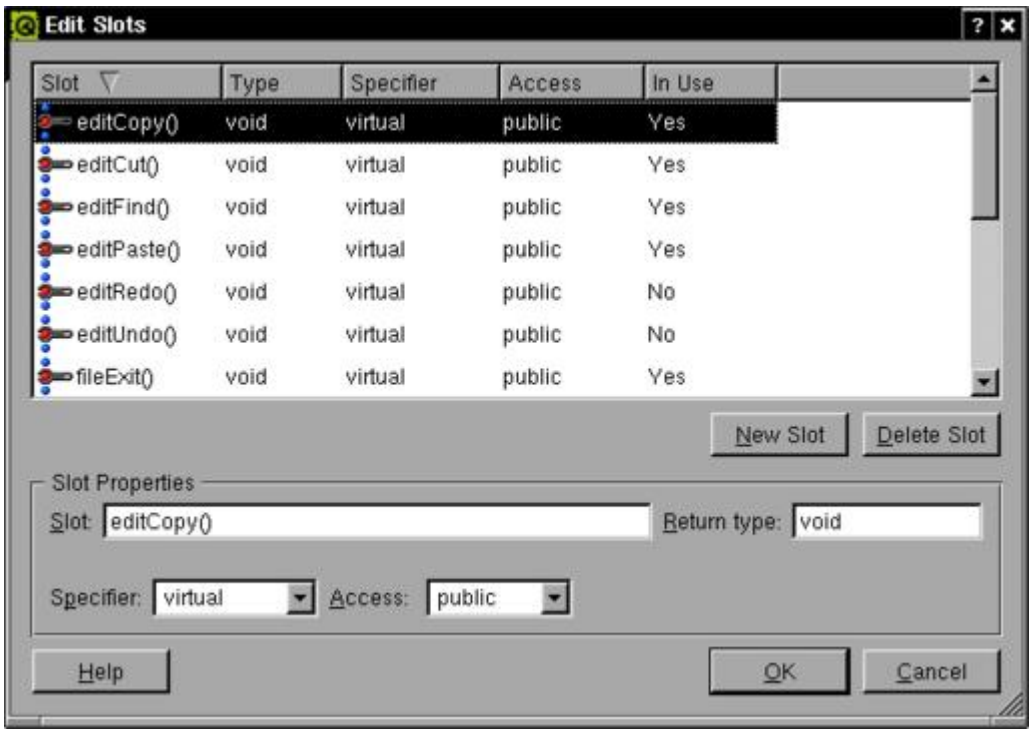


Figure 7. A new function skeleton comes into being.

Now we can connect the fileCloseAction's activated() signal to ljeditor::fileClose() in the Edit Connections dialog provided by the Action Editor. As this action should be available via menu only, we simply drag and drop it into the File menu of the ljeditor form.

Adding Some Code

Closing a widget in Qt is easy: every Qt widget inherits a close() function from the mother of all Qt widgets, QWidget. This is not really much code, so it would be nice if we could fill the fileClose() slot with this one line.

A right-mouse click on the ljeditor form opens a context menu. The choice of its Source... entry does the trick. A code editor window appears and allows us to fill in the one-liner (see Figure 8):

```
void ljeditor::fileClose()
{
     close();
}
```

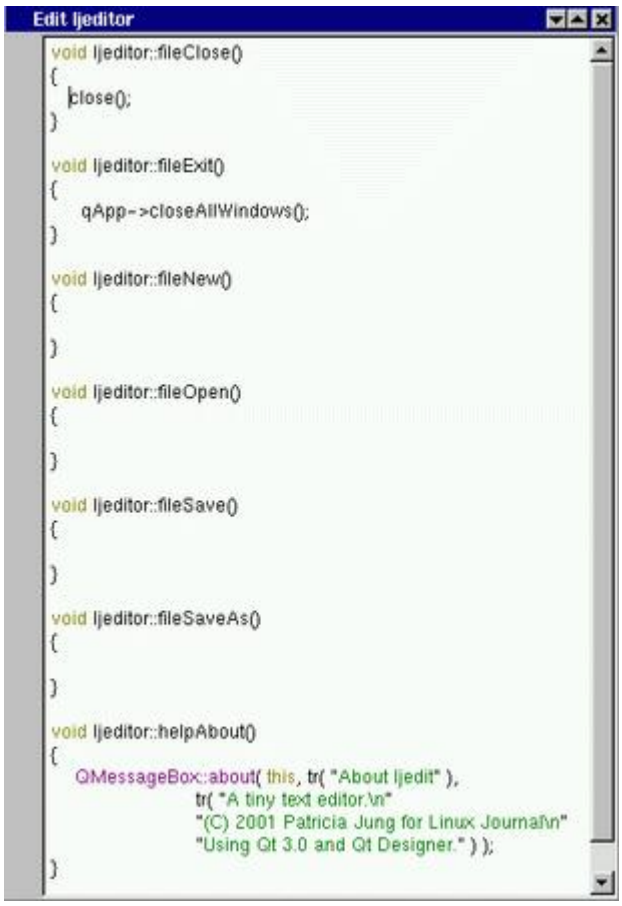


Figure 8. Simple code lines are easily added to the slot skeleton.

So why not fill the fileExit() slot as well? To quit the entire application, the application object's closeAllWindows() function is called:

```
void ljeditor::fileExit()
{
```

```
        qApp->closeAllWindows();
    }
```

As the Designer usually does not deal with QApplication objects itself, qapplication.h (which provides the qApp proxy for the real application object) is not included by default, and the code generated from the ui description would not compile.

Fortunately, the Object Explorer allows us to include additional header files. By default it shows the Widgets tab (see Figure 4, bottom left), but we need the Source tab right now. Right-clicking the empty Includes (in Implementation) folder allows us to add a header file (New). Don't forget the <> brackets around a global include like <qapplication.h> (Figure 9).

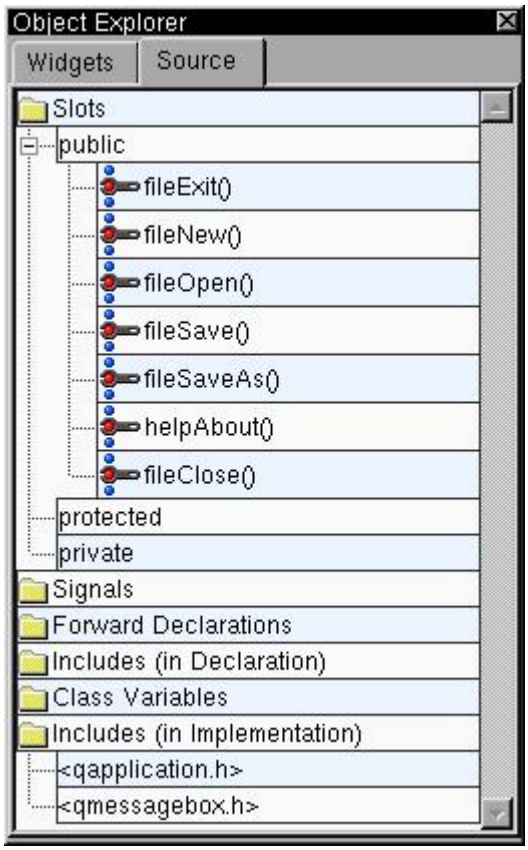


Figure 9. The Object Explorer allows one to include additional header files.

Another slot we can fill with life without being afraid of getting too many compilation errors is helpAbout(). It is called when the user opens the About... entry in ljeditor's Help menu and simply pops up a message box with the caption About ljedit and some information about the program. By surrounding all text strings with tr() we make sure that the program can be localized painlessly, a task we will fulfill in Part II. For example:

```
void ljeditor::helpAbout()
{
    QMessageBox::about( this, tr( "About ljedit" ),
        tr( "A tiny text editor.\n"
        "(C) 2002 Patricia Jung for Linux Journal\n"
```

```
            "Using Qt 3.0.4 and Qt Designer." ) );
    }
```

To be able to use QMessageBox::about(), we have to include <qmessagebox.h> the same way we did with <qapplication.h>. We add the remaining functionality in a subclass next time. Thus, all we have left to do with the Designer is clean up the new GUI.

Important Cosmetics

Before we leave the Designer for good, we check that none of the assigned keyboard shortcuts has been used twice. This is done by choosing Check Accelerators from the Edit menu.

Additionally, we clean up all the ljeditor slots that we weren't going to implement anyway or that we used TextEdit slots for instead. By choosing Slots from the Edit menu (one way to do it), we open the dialog that allows us to mark editUndo() and remove it with a mouse click on Delete Slots (Figure 7). The same fate applies to editRedo(), editCut(), editCopy(), editPaste(), editFind(), helpIndex(), helpContents() and filePrint().

Furthermore, we check the GUI preview for separators that don't fit. As we decided against using the editFindAction, a single separator can be found at the end of ljeditor's Edit menu: once upon a time there was a Find entry below. To erase the separator, right-click on it in the form and choose Delete Item. The same applies to one of the two separators above the Exit entry in the file menu.

All GUI elements are in their place—time to let Qt adjust the widget proportions. Once again, we choose the entire form and select Lay Out Vertically from Layout in the main menu. If a user adjusts the application window size now, the TextEdit widget will follow. If we wish to place widget elements differently onto the form, we need to break the layout first.

Last but not least, we want this GUI to be stamped as ours. To do this we fill in the Author name and a description in the dialog raised by choosing Form Settings from the Edit menu (see Figure 10). Here it is possible to decide whether we want to store the icons in a subdirectory of our project directory or whether we include them directly in the user-interface description.

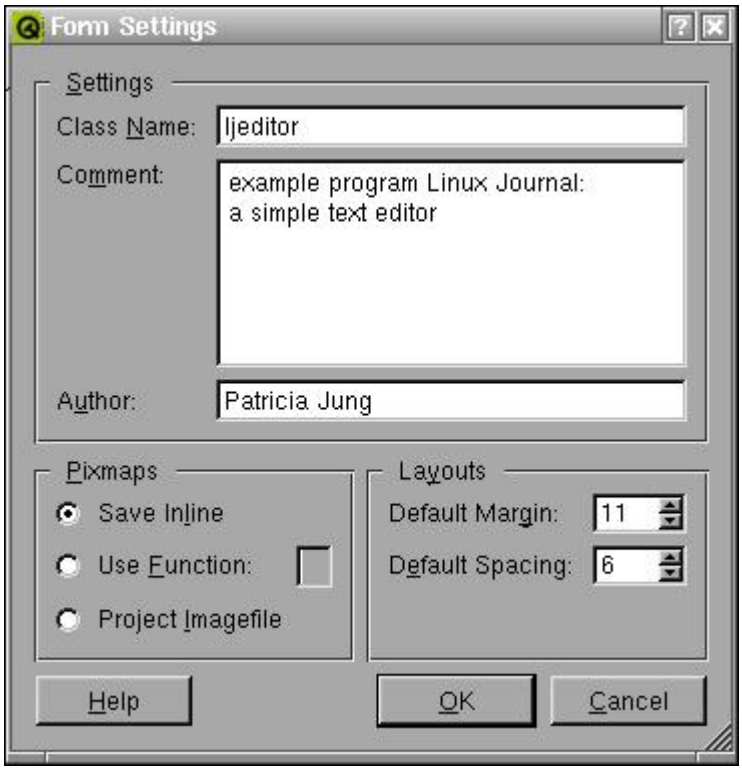


Figure 10. Let the world know who designed this GUI.

A final selection of Save all from the File menu and an XML file with the user-interface description (best named after the relevant class, e.g., ljeditor.ui) along with ljeditor.ui.h, the file storing the code we typed into the code editor, becomes part of the project.

Patricia Jung (trish@trish.de) has a background as a system administrator, technical writer and editor, and as such is happy to have the privilege of dealing with Linux/UNIX exclusively.

Archive Index Issue Table of Contents

Advanced search

Bring an Atomic Clock to Your Home with Chrony

Fred Mora

Issue #101, September 2002

Among all the techno-toys that make a true geek salivate, few are as cool as an atomic clock.

Here is a device that finally provides what generations of scientists have dreamed about: an ultra-precise time reference, a timekeeping piece of incredible accuracy. These are not simple gadgets; some physics experiments, such as the verification of gravitation theories, require measurement of very small time intervals.

The National Institute of Standards and Technology (NIST) has a lab in Boulder, Colorado, devoted to running atomic clocks and providing official US time. In this lab, the NIST-F1 cesium fountain atomic clock provides a time reference with a precision of 2 × 10-15 (two parts per millionth of a billionth) by counting the vibrations of cesium atoms at about 9GHz. An even better clock is in the works. It will measure the resonance of a single mercury ion at about 100,000 times that frequency, and it will provide a thousand-fold increase in precision.

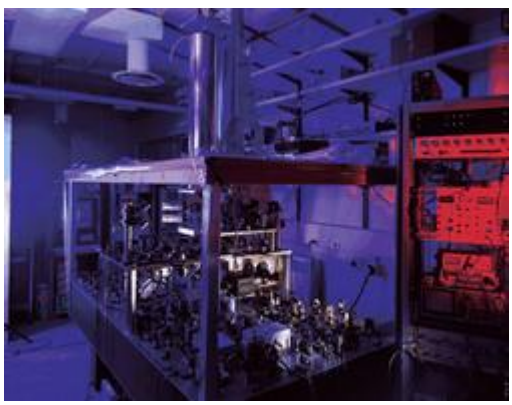


Figure 1. Atomic Clock

The sacred duty of true precision-obsessed geeks is now clear. They simply have to synchronize the real-time clock of their Linux machine(s) with such an insanely precise clock.

Of course, you cannot simply go to a computer store and buy an atomic clock. (Not that I didn't try—sheesh, the brazen gall of that sales guy, trying to saddle me with a radium dial alarm clock.) The next best thing is a radio-synchronized clock, and a variety of models are available. They can be connected to the serial port of a PC and provide time signals synchronized on the NIST clock.

But why buy hardware when well-designed, free software would do the trick? The Network Time Protocol (NTP) has been created to synchronize computers and distribute time references across networks. An NTP server keeps time as close to the official reference as possible. Remote NTP clients query these servers and sync the local real-time clock (RTC) of the machine. This timekeeping is a complex problem due to the nature of distributed computing. Propagating packets between machines takes a nonzero, variable time. Various correction schemes are integrated in NTP to take variable latency into account.

Why Chrony?

There are several NTP clients and servers available for Linux. The simplest way of using NTP would be to fire up a program, such as xntpd, and point it to an NTP server. However, this program and most other NTP clients work best when they are connected to the Internet continuously. Unfortunately, an intermittent connection through a modem is still the way most homes access the Internet.

That's where chrony comes in. Chrony is a program that explicitly supports intermittent connections. It is comprehensive but a tad intimidating, so we'll walk through an installation and configuration for the most common case: a home user with a modem connection.

Chrony is composed of chronyc, a text-based client program; chronyd, an NTP server running as a dæmon in the background; and various configuration files. To interact with the chronyd dæmon (chronyd), you run the chronyc client and issue commands.

Downloading and Installing

Some Linux distributions include a version of chrony. Chances are that this version is an older one, e.g., 1.15 or less. In that case, you can uninstall the chrony package before installing the new version.

First, download the chrony tarball from its home page (see Resources). At the time of this writing, the current version is 1.16.1. It is composed of the 1.16 version completed by a patch to 1.16.1. We extract the source from the tarball and apply the patch:

```
tar -zxvf chrony-1.16.tar.gz # extracts source
cd chrony-1.16  # dir created from tarball
```

```
gunzip < ../chrony-1.16-1.16.1-patch.gz | patch -p1
patching file NEWS
patching file configure
patching file rtc_linux.c
patching file version.txt
```

The program uses a **configure** script, which makes customization a snap. The only option that you need to specify manually is the installation directory, with the --prefix option. By default, chrony will install the client chronyc into /usr/local/bin and the dæmon chronyd into /usr/local/sbin. It is the equivalent of:

```
# In the same chrony-1.16 dir as before
./configure --prefix /usr/local
```

Once you have run **configure**, you might want to clean up the source a tad before running **make**. Why? Because the source comes with a few syntactic gotchas that make the GCC preprocessor complain. If you run make right away, you'll end up with plenty of warnings such as:

```
warning: extra tokens at end of #endif directive
```

Nothing is broken, but it's easy to get it to compile cleanly. Edit the files regress.h, reports.h and rtc_linux.h. The last line is an #endif statement followed by a constant name. You need to comment out that name. For instance, in report.h, change:

```
#endif GOT_REPORTS_H
```

to:

```
#endif /* GOT_REPORTS_H */
```

and chrony will compile like a charm.

Now, do:

```
# In the same chrony-1.16 dir as before
make
su root  # You need to be root to install
install
```

The next step is to make sure that chronyd starts up at boot time. If your distribution came with an older version of chrony, then you are all set; just make sure that the newer version was installed in the same location as the old one. Otherwise, there are several methods. The simplest is to add a paragraph supplied by the chrony doc in your /etc/rc.d/rc.local.

Stratum Conundrum

Now, it's time to configure chrony. Because you want to sync your machine's clock on an NTP server, you need to pick one. Actually, you'll need to pick a few different servers in case one of them is unreachable. See the URL for the list of NTP servers in Resources.

The list separates the NTP servers into several strata. Now, what's a stratum? We are not talking about a geological layer of rock here. Think onion rings instead. Stratum 1 is composed of servers that are directly synced with an atomic clock. Stratum 2 is a set of NTP servers that are fed timestamps by stratum 1 and so on. Keep away from stratum 1 unless you run a physics lab or your private network has hundreds of machines. Stratum 1 machines should be reserved for distributing time references to secondary servers or to machines that cannot settle for the few microseconds of imprecision incurred by using stratum 2. For our purpose, choosing servers in stratum 2 or even 3 will be perfectly fine.

Notice that the administrators of some NTP servers require you to e-mail them if you want to sync with their machine. Please do so. If a thousand home machines suddenly start requesting timestamps from a poor university NTP server, the administrators need to know that it's actually for syncing clocks and not some new form of flood attack.

Ideally, you should pick an NTP server that is not too far away from your machine (IP-wise). This is not always the same as geographically. A rule of thumb is to check the output of **traceroute**, which lists the systems traversed by packets between your machine and the destination. For instance, since I am a New York dweller, I picked the following machines: ntp.ctr.columbia.edu from Columbia University, clock.psu.edu from Pennsylvania State University and ntp0.cornell.edu from Cornell University (send an e-mail to pgp1@cornell.edu if you use this server).

Become root and edit your /etc/chrony.conf file to add these definitions:

```
server ntp.ctr.columbia.edu offline
server
server ntp0.cornell.edu offline
```

Of course, replace these server names with the names of NTP servers located close to you. Note that the servers are declared off-line. Most modem-connected machines do not start a connection at boot. So when chronyd starts up, it should not start querying servers. Also, note that the chrony doc insists that you should use the TCP/IP numerical address of the NTP servers to alleviate a dependency on DNS. Well, the administrators of most NTP servers want you to use only the DNS-declared names so that they retain the ability to move the servers around. Besides, your modem connection hopefully can reach your ISP's DNS, so I recommend that you use the NTP server names in the chrony.conf file.

Passwords

The NTP protocol supports packet authentication. After all, if you run a company, you don't want wrongdoers to set your machines' clocks to an arbitrary time. Financial records with an incorrect timestamp can throw your auditors into a loop.

Chrony supports that authentication with a simple password scheme. You can define several chrony users identified by a number and give them different passwords. The relevant statements in the chrony.conf file are:

```
commandkey 9
keyfile /etc/chrony.keys
```

This specifies that this machine uses key number 9 to be read from the passwords stored in /etc/chrony.keys. The latter contains, for instance:

```
9 zack
```

Zack is the name of my cat. Before I started using chrony, the beast was the closest thing to a precise clock I had in the house. Every morning at 7:30, he meows pathetically until I feed him—including weekends. He quickly became very good at ducking pillows.

Also, chrony needs to store the rate at which your machine's clock deviates, or drifts, from the NTP server time. Specify a location for the drift file with:

```
driftfile /etc/chrony.drift
```

This way, chrony does not have to accumulate measurements and recalculate the drift every time you start it.

You might have several computers in your home. In that case, it's a good idea to sync their clocks, too. By default, chronyd acts strictly as an NTP client with respect to the servers you define in the server statements. But you can set chronyd to act like a server for your own subnetwork. Just add an allow statement to your chrony.conf, and specify either some machines or a subnetwork. For example, my home machines' Ethernet cards have addresses in the (nonroutable) 192.168 subnet, and the machine acting as a server has the statement:

```
allow 192.168
```

in its chrony.conf file. This way, other machines in my home can sync with my server (address 192.198.0.1) by running chronyd with this simple configuration file:

```
server 192.198.0.1
keyfile /etc/chrony.keys
commandkey 9
driftfile /etc/chrony.drift
```

To summarize, the chrony.conf file of your modem-connected machine is given below. Note: replace the NTP servers in this example with the ones that you pick in the list of NTP servers (see Resources):

```
server ntp.ctr.columbia.edu offline
server
server
commandkey 9
keyfile /etc/chrony.keys
driftfile /etc/chrony.drift
```

Running the Client

So now that chrony is installed, verify that chronyd runs in the background (start it if necessary). Remember that the configuration file specifies (with the offline keyword) that chronyd should not query the servers without your permission. Start your modem connection, verify that you are connected to your ISP and then start the chronyc client. Figure 2 shows a sample session.

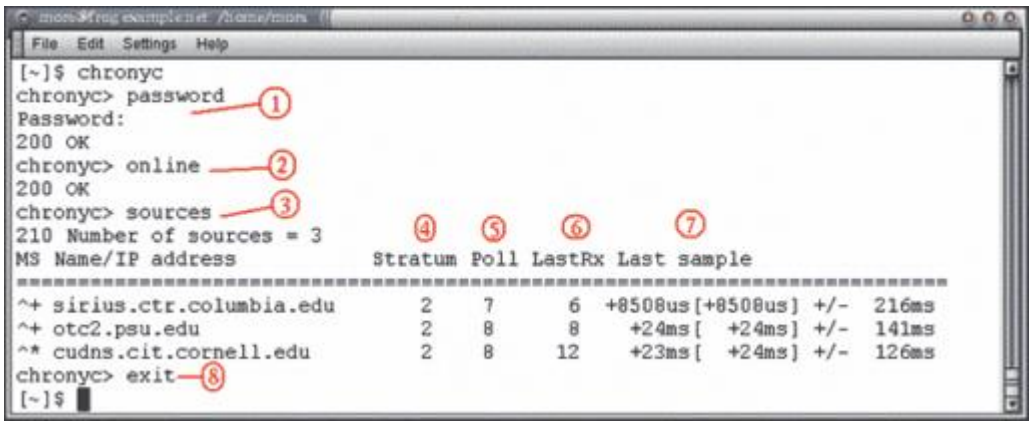


Figure 2. Sample chronyc Session:

1. Password command—notice that your password doesn't echo.
2. The online command tells chronyd to start using the NTP servers.
3. The source command lists the NTP servers known by chronyd: ^ means a server, * indicates the source to which chronyd is currently synchronized and + indicates other acceptable sources.
4. Stratum 2 is good enough.
5. The base-2 logarithm of the number of seconds between two polls of the server: 7 is 128 sec, 8 is 256.
6. Time since the lastsample was received from the source (in seconds unless you see m, h, d or y for minutes, hours, days or years).
7. Offset measurement from the last sample. First comes the original measurement, then the actual offset between brackets, then the margin of error.

The first command you enter in chronyc should be the password command. Then, order the dæmon to start talking to the NTP servers with the online command. List the NTP servers (**sources -v**, which is the verbose form of the sources command). See the tilde (~) in the second column? It says that the server cannot be used yet. It's too early; the dæmon needs a couple of minutes to accumulate timestamps and make sure the responses of the NTP servers aren't delirious. By some cosmic quirk, the difference between my machine's clock and the NTP timestamps happens to be 42 seconds (all hail Douglas Adams!).

Wait a moment and issue another sources command. After a while, you'll see that one of the servers has been selected by chronyd (a star appears in the second column) and that the offset of your machine is decreasing:

```
^* cudns.cit.cornell.edu
2    6     54  +2999ms[+2999ms] +/- 3653ms
```

Chrony slowly accelerates or slows your clock to make it reflect the NTP time. So over the course of a few minutes, by gradual correction, any offset will disappear.

Other useful commands include:

- tracking: shows how your system clock is doing, that is, how fast or slow it is with respect to NTP sources.
- sourcestats -v: shows what chronyd thinks of the sources from the measurements it has obtained so far.
- makestep: immediately sets your system's clock to the NTP time instead of gradually skewing the clock. This is the equivalent of setting the time. Some versions of X11 can freeze if you set the time back brutally.

Finally, remember to issue an offline command in chronyc before you disconnect your modem. Otherwise, chrony will believe the source it has picked has become unreachable and frantically will try to pick a new one.

Automated Sync

As you can guess, chronyc begs for automated operation. You can easily create two little scripts that will set chrony on-line and off-line. The on-line script:

```
#! /bin/sh
# This script is called after connect
/usr/local/bin/chronyc <<EOF
password  zack
online EOF
```

should be called after the modem connection has been established, and the off-line script:

```
#! /bin/sh
# This script is called before disconnect
/usr/local/bin/chronyc <<EOF
password zack
offline
EOF
```

should be called right before you disconnect.

If you use a special dialer, check if it has options to allow post-connect and pre-disconnect commands. I am using the ATT Global Network dialer, and it allows me to put such scripts in its /opt/attdial/bin. If you are using the plain vanilla PPP, you can insert the on-line script in the /etc/ppp/ip-up file and the off-line script in /etc/ppp/ip-down. Some distributions want you to leave ip-up and ip-down alone and modify only ip-up.local and ip-down.local (check to see if these files exist).

Conclusion

I found chrony the ideal tool for syncing my machine through a modem connection that is only up a few hours a week. I'd like to thank chrony's author, Richard Curnow, who sent me valuable tips and answered many questions quickly.

Resources

email: fmora@us.ibm.com

Fred Mora has been a UNIX system administrator and developer since 1990. He has published and coauthored several books and technical manuals. He is doing his best to lose the rest of his sanity by tinkering with Linux and writing more books, with the encouragement of his techie wife. He works at IBM.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

CVS homedir

Joey Hess

Issue #101, September 2002

Joey shows you how to keep track of everything with CVS.

I keep my life in a CVS repository. For the past two years, every file I've created and worked on, every e-mail I've sent or received and every config file I've tweaked have all been checked into my CVS archive. When I tell people about this, they invariably respond, “You're crazy!”

After all, CVS is meant for managing discrete bodies of code, such as free software programs that are worked on and available to a lot of people or in-house projects that are collaboratively developed by several employees. CVS has a reputation of being a pain to deal with, and it has a lot of crufty bits that regularly drive users up the wall, like its mistreatment of directories. Why inflict the pain of CVS on yourself if you don't have to? Why do it on such a scale that it affects nearly everything you do with your computer?

I get three major benefits from keeping my whole home directory in CVS: home directory replication, history and distributed backups. The first of these is what originally drove me to CVS for my whole home directory. At the time, I had a home desktop machine, two laptops and a desktop machine at work. Rounding this out were perhaps 20 remote accounts on various systems around the world and many systems around the workplace that I might randomly find myself logging in to. I used all of these accounts for working on the same projects and already was using CVS for those projects.

I'm a conservative guy when it comes to my computing environment (I've used the same wallpaper image for the past five years), and at the same time I'm always making a lot of little tweaks to improve things. Whenever I go to work and something wasn't just like I had tweaked it the night before, I'd feel a jarring disconnect, and annoyingly copy over whatever the change was. When I sat down at some other system at work, to burn a CD perhaps, and found a bare Bash shell instead of the heavily customized environment I've built up

over the past ten years, it was even worse. The plethora of environments, each imperfectly customized to my needs by varying degrees, was really getting on my nerves. So one day I cracked and sat down and began to feed my whole home directory into CVS.

It worked astonishingly well. After a few weeks of tweaking and importing I had everything working and began developing some new habits. Every morning (er, afternoon) when I came into work, I'd **cvs up** while I read the morning mail. In the evening, I'd **cvs commit** and then update my laptop for the trip home. When I got home, I'd sync up again, dive right back into whatever I'd been doing at work and keep on rolling until late at night—when I committed, went to bed and began the cycle all over again. As for the systems I used less frequently, like the CD burner machine, I'd just update when I got annoyed at them for being a trifle out of date.

It only took a few more weeks before the advantage of having a history of everything I'd done began to show up. It wasn't a real surprise because having a history of past versions of a project is one of the reasons to use CVS in the first place, but it's very cool to have it suddenly apply to every file you own. When I broke my .zshrc or .procmailrc, I could roll back to the previous day's or look back and see when I made the change and why. It's very handy to be able to run **cvs diff** on your kernel config file and see how **make xconfig** changed it. It's great to be able to recover files you deleted or delete files because they're not relevant and still know you've not really lost them. For those amateur historians among us, it's very cool to be able to check out one's system as it looked one full year ago and poke around and discover how everything has evolved over time.

The final major benefit took some time to become clear. Linus Torvalds once said, “Only wimps use tape backup: *real* men just upload their important stuff on FTP and let the rest of the world mirror it.” I'm not a real enough man to upload my confidential documents to ftp.kernel.org though, so I've been wimping along with backups to tape and CD and so on. But then it hit me: take, for example, one crucial file, like my .zshrc or sent-mail archive: I had a copy of that file on my work machine, and on my home machine, and on my laptop and several other copies on other accounts. There was another copy encoded in my CVS repository too.

I'm told that the best backups are done without effort—so you actually do them—and are widely scattered among many machines and a lot of area so that a local disaster doesn't knock them out. They are tested on a regular basis to make sure the backup works. I was doing all of these things as a mere side effect of keeping it all in CVS. Then I sobered up and remembered that a dead CVS repository would be a really, really bad thing and kept those wimpy

backups to CD going. But the automatic distributed backups are what keep me sleeping quietly at night. Later, when I left that job, the last thing I did on my work desktop machine was: **cvs commit ; sudo rm -rf /**. And I didn't worry a bit; my life was still there, secure in CVS.

A full checkout of my home directory with all the trimmings often runs about 4GB in size. A lot of that will be temporary trees in tmp/ and rsynced Ogg Vorbis files (so far, I have not found the disk space to check all of them into CVS). My CVS repository currently uses less than 1GB of space, though it is steadily growing in size. I keep some 13,000 files in CVS, and so a full CVS update of my home directory is a sight to see and takes a while.

These days I'm often stuck behind a dial-up connection, and I mostly just use one laptop, so I might go days between CVS updates. Other better-connected systems have automatic CVS updates done via cron each day. I **cvs commit** whenever I want to make a backup of where I am in a file or when I am at the point of releasing something. I still also do a full commit of my home directory every day or so. I confess that some of my CVS commit messages are less than informative—“foo” has been used far too many times on some classes of files. I even do some automatic CVS commits; for example, my mailbox archives are committed by a daily cron job.

There are other benefits of course. I attend many tradeshows and other events that require me to sit down at some computer out of the box, use it for an hour or a day and never see it again. I can check out the core of my CVS home directory in about five minutes, and after that it is just as comfortable as if I'd SSH'd home and was doing everything there. I even get my whole desktop set up in that five minutes. In a chaotic tradeshow environment, there is nothing more reassuring than having your familiar computer setup at your fingertips as you demo things to the hordes of visitors.

Keeping your home directory in CVS is not all fun though. Anyone who's used CVS in a large project probably has had to resolve conflicts engendered by two people modifying the same file. At least you can curse the other guy who committed the changes first while you deal with this annoying task. Most of you have probably *not* had to resolve conflicts between the file you modified at home and at work, then cursing at yourself.

Then there are CVS's famous problems: poor handling of directories and binary files. The nearly nonexistent handling of permissions, which is not a big deal in most projects but becomes important when you have a home directory with some public and some private files and directories in it. A slow, bloated protocol, hindered even more by the necessity of piping it all over SSH; the pain of trying to move a file that is already in CVS, or much worse, a whole directory

tree, again hits you especially hard when you're using CVS for the whole home directory. And those damn CVS directories are always cluttering up everything. I've developed means of coping with all of these to varying degrees, but like many of us, I'm hoping for a better replacement one day (and dreading the transition).

Perhaps it's time that I get down to the details of how I organize my home directory in CVS. I've always managed my home directory with an iron hand, and CVS has just exacerbated this tendency. Let's look at the top level:

```
joey@silk:~>ls
CVS/  GNUstep/  bin/  debian/  doc/  html/  lib/
mail/  src/  tmp/
```

Yes, that's it. Well, except for the 100-plus dot files. Most people use their home directory as a scratch space for files they're working on, but instead I have a dedicated scratch directory, the tmp directory, which I clean out irregularly. In general, when I start a new file or project, I will be checking it into CVS soon, so I begin working on it in the appropriate directory. This document, for example, is starting its life in the html directory and will be checked into CVS soon to live there forever. Of course, sometimes I goof up and then I have to resort to the usual tricks to move files in CVS. And so the first rule of CVS home directories is it pays to think before starting and get the right filename and location the first time. Don't be too impatient to check in the file.

CVS is a great way to ensure that you have a nice, clean, well-managed home directory. Every time I **cvs update** it will helpfully complain to me about any files it doesn't know about. Of course, I make heavy use of .cvsignore files in some directories (like tmp/).

If I go to another machine, the home directory looks pretty much the same, though various things might be missing:

```
joeyh@auric:~>ls
CVS/  GNUstep/  bin/  tmp/
```

I use this machine for occasional specific shell purposes. I don't administer the system, so I don't want to put private files there. The result is a much truncated version of my home directory. It's perfectly usable for everything I normally do on that machine, and if I want to, say, work on this document there at some point, I can just type **cvs co html** and a password and be on my way.

The way I make this partial-checkouts system work is by using CVS modules and aliases. I have modules defined for each of the top-level directories and for the home directory (dot files) itself. For example, the entry in my CVSROOT/modules file for the stripped-down version of my home directory looks like this:

```
joeyh -u cvsfix -o cvsfix joey-cvs/home &bin
```

For more complete home directories, I use this instead:

```
joey -u cvsfix -o cvsfix joey-cvs/home &src &doc
&debian &html &lib &.hide &bin &mail
```

Notice the .hide module. It results in a ~/.hide directory when I check it out. This directory is where I put the occasional private file that I don't want to appear in home directories—like the one on auric—that are on systems not administered by me. The files in .hide get hard-linked to their proper locations if .hide is checked out, so I can put confidential dot files in there and only check those dot files out on trusted systems. I also have, for example, my Mozilla cookies file in .hide.

It's important to distinguish between such files that I need to put in .hide and the entire set of private directories, like my mail directory. Yes, I keep my mail in CVS (except for just-arrived spooled mail, which I keep synced up with a neat little program called isync that is smarter about mail than CVS is). But it's all in its own mail/ directory, so I can omit checking that directory out to systems that I don't trust with my mail or that I don't want to burden with hundreds of megabytes of mail archives.

While I'm discussing privacy issues, I should mention that I make some bits of my home directory completely open to the public. This includes a lot of free software in debian/ and src/, and some handy little programs in bin/. This is accomplished by permissions. I have to make sure that most directories in the repository (or at least the top-level directories like mail/) are mode 700, so only I can access them. Other top-level directories, like bin/, are opened up to mode 755. This allows anonymous CVS access and browsing at cvs.kitenet.net/joey-cvs/bin/.

This leads to the second rule of CVS home directories: don't import $HOME in one big chunk; break it up into multiple modules. The structure of your repository need not mirror the structure of your actual home directory. Modules can be checked out in different locations to move things around and control access on a per-module level. There's a layer of indirection there, and such layers always make things more flexible and more complex.

Some of the projects I work on have their own CVS repositories that are unconnected to my big home directory repository. That's fine too; I simply check them out into logical places in my home directory tree as needed. CVS can even be tweaked to recurse into those directories when updating or committing.

Another thing to notice in those lines from my modules file is the use of **-u cvsfix** to make the cvsfix program run after CVS updates. That program does a lot of little things, including ensuring that permissions are correct, setting up the hard links to files in .hide and so on.

One last thing to mention is the issue of heterogeneous environments and CVS. Most of my accounts are on systems running varying versions of Debian Linux on a host of different architectures, but there are accounts on other distributions, on Solaris and so forth. Trying to make the same dot files work on everything can be interesting. My .zshrc file, for example, goes to great pains to detect things like GNU ls, deals with varying zsh versions, sets up aliases to the best available editor and other commands and so on. Other programs, like .xinitrc, check the host they're running on and behave slightly (or completely) differently. I've even at one point had a .procmailrc that filtered mail differently depending on hostname, though the trick to doing that is lost somewhere in one of the innumerable versions stored in my repository. I've even resorted in a few places to files with names of the form filename.hostname—cvsfix finds one matching the current host and links it to the filename. Branches are also a possibility, of course, but despite my heavy use of CVS, I still find some corners of it a black art.

Well I guess that's it. I'd be happy to hear from anyone else who keeps their home directory in CVS, especially if you have some tricks to share. In the future I'd like to try checking /etc into CVS too, and if you've successfully done this, I'd love to talk with you. Now I'm off to commit this file.

Joey Hess (joey@kitenet.net) is a longtime Debian developer who lives on a farm in Virginia. He enjoys finding new and unlikely places from which to commit code wirelessly to CVS.

Archive Index Issue Table of Contents

Advanced search

Linux Multimedia with PD and GEM: a User's Report

Dave Phillips

Issue #101, September 2002

A look at the possibilities of Pd and GEM for Linux-based audio and video.

As a multimedia-capable platform Linux has seen terrific growth over the past few years. At the system level, a simple kernel patch can now improve scheduler efficiency and bring performance latencies down to an incredible three milliseconds or less, well within the acceptable range for professional audio and video applications. Given the low-latency patch (along with some other fine-tuning), we can now consider the availability of applications capable of utilizing this enhancement.

Along with performance artist Chris Spradlin, I am currently working on a multimedia presentation that combines video playback/processing, still-image and video projection, and various forms of audio capture, playback and transformation. Controlling the interplay of the different media poses a considerable challenge, particularly because we want to do everything in real time and in Linux. Happily, I have found an excellent solution to our dilemma: Miller Puckette's remarkable Pd.

From the Simple

Pd (pure data) is an audio synthesis/processing environment similar to the famous Max (and its Java offspring jMax). These environments employ a neat scheme of graphically patching various simple components (such as signal generators/modifiers and their control objects) into complex sonic networks.

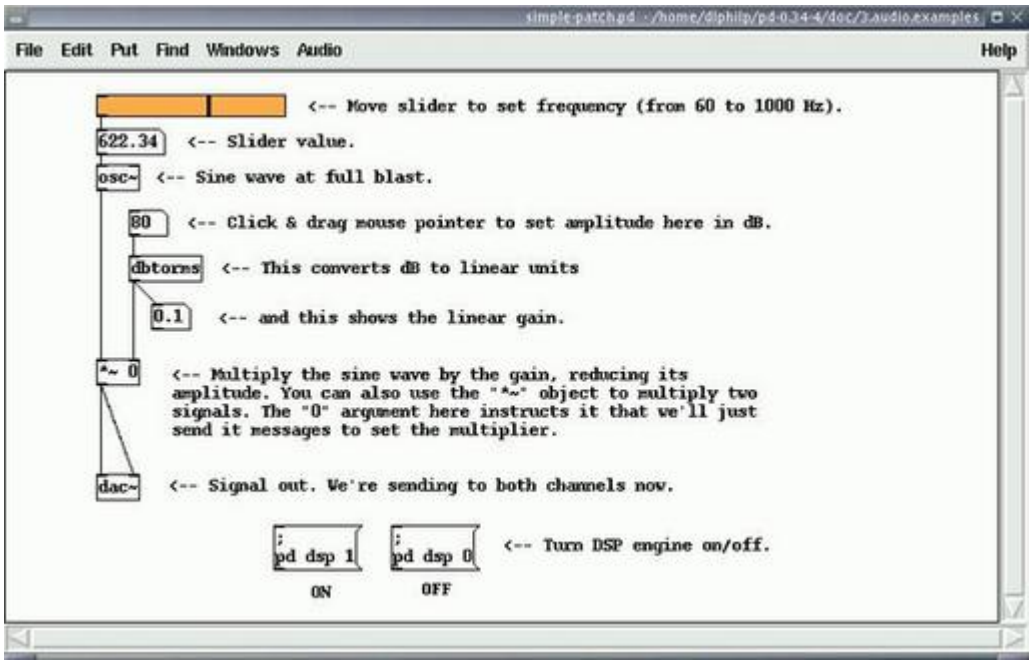


Figure 1. A Simple Pd Patch

Figure 1 demonstrates Pd's basic principles: the osc~ signal generator creates an audio waveform (a cosine wave), the slider controls the frequency (pitch) of the waveform and the network around the dbtorms object modifies the amplitude (volume) of the generated signal. Finally, the modified signal is sent to the dac~ object (the digital-to-analog converter) and the results are heard through the audio system.

Pd includes a variety of ready-made objects for signal generation and processing, and if this kind of synthesis patching were all Pd could do, it would still be an impressive audio environment. However, thanks to Mark Danks' wonderful GEM OpenGL-based graphics library, Pd also can manipulate video and image parameters in real time. Pd's flexibility permits arbitrary connection and control between its audio and video streams, creating a powerful environment for controlling and coordinating multimedia presentations.

In order to use Pd with GEM, you must invoke it via the $HOME/.pdrc file or with a command string similar to this one:

```
pd -rt -lib /home/dlphilp/gem-0.87_2/Gem
```

The -rt option prioritizes Pd's performance to real-time status. When coupled with a low-latency kernel, Pd's performance is quite acceptable for live shows and other real-time circumstances. You'll want all the help you can get when you're running Pd with the GEM library; the kernel latency patch is a godsend, but you'll still need a hardware-accelerated OpenGL installation to make the best use of GEM.

Note: the test system for this article included an 800MHz Duron CPU, 256MB RAM and a Voodoo3 AGP video card. The Linux kernel version was 2.4.5, patched for low latency; the video subsystem was XFree86 4.0.1. Certain operations in GEM are very CPU-intensive, and I would qualify the Voodoo3 as the low end of acceptable video boards for the Pd/GEM alliance. The audio system included a Sound Blaster Live! sound card running under the ALSA 0.9.0beta10 driver. Note also that Pd version 0.34-4 (stable) was used along with IOhannes Zmoelnig's beta version of GEM 0.87. Previous incarnations of GEM do not include the Linux versions of the pix_movie and pix_film objects needed for the real-time video manipulations described here.

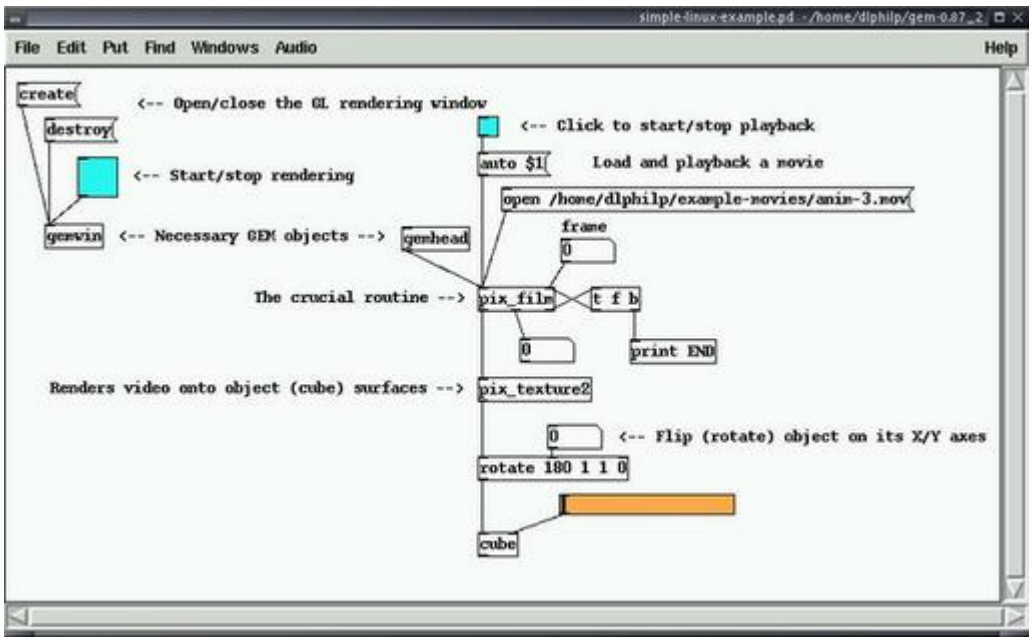


Figure 2. A Simple Pd/GEM Patch

In Figure 2 we see the basic structure of a simple Pd patch utilizing the GEM library functions. Note that the gemwin and gemhead objects are required for all other GEM-related actions. This patch provides the mechanisms for loading a movie (anim-3.mov in this case) and playing it back while rendering it to the surfaces of a cube. The cube size is controlled by the slider movement, and the film can be started and stopped by clicking on the smaller Bang button (one of the two cyan-colored boxes). Figures 3-5 demonstrate this patch in action. The first snapshot (Figure 3) illustrates the simplest rendering, with one face of the cube displayed and expanded to fit most of the rendering window. Figure 4 shows how the rotate object manipulates the cube, and Figure 5 displays the curious effect that occurs when the slider value exceeds the rendering window size. Of course the still images can't convey the effect of the movie playing upon the surfaces of the animated cube, but trust me, it's very cool.

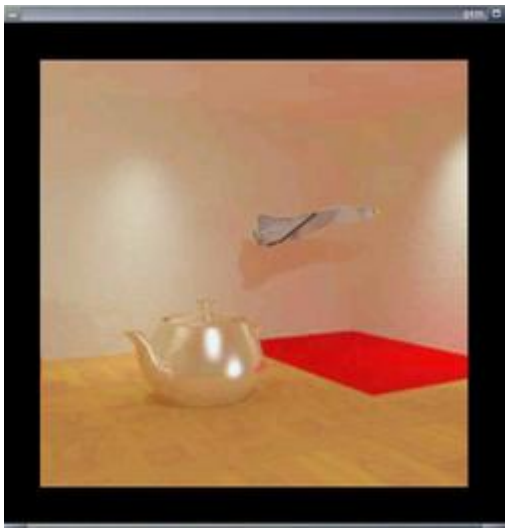


Figure 3. Default Video Display

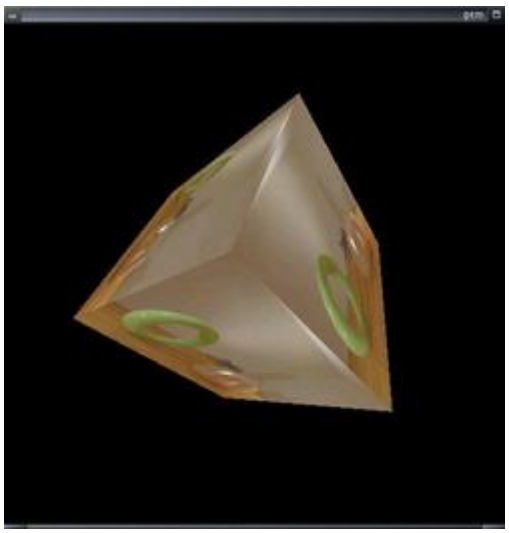


Figure 4. Video on a Cube

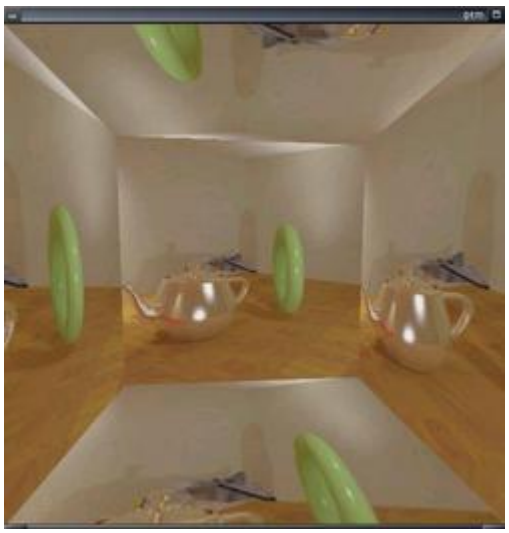


Figure 5. Exploded Video Display

To the Complex

Now let's look at the possibility of combining our two example patches. Using straightforward cut/copy/paste editing, we easily can copy one patch's contents into another and start having some serious fun. Pd truly lives up to the promise of its name: data is purely data here, any data stream can be routed anywhere within a patch (with some restrictions), and we easily can set up a system in which one set of controllers simultaneously controls audio and video parameters.

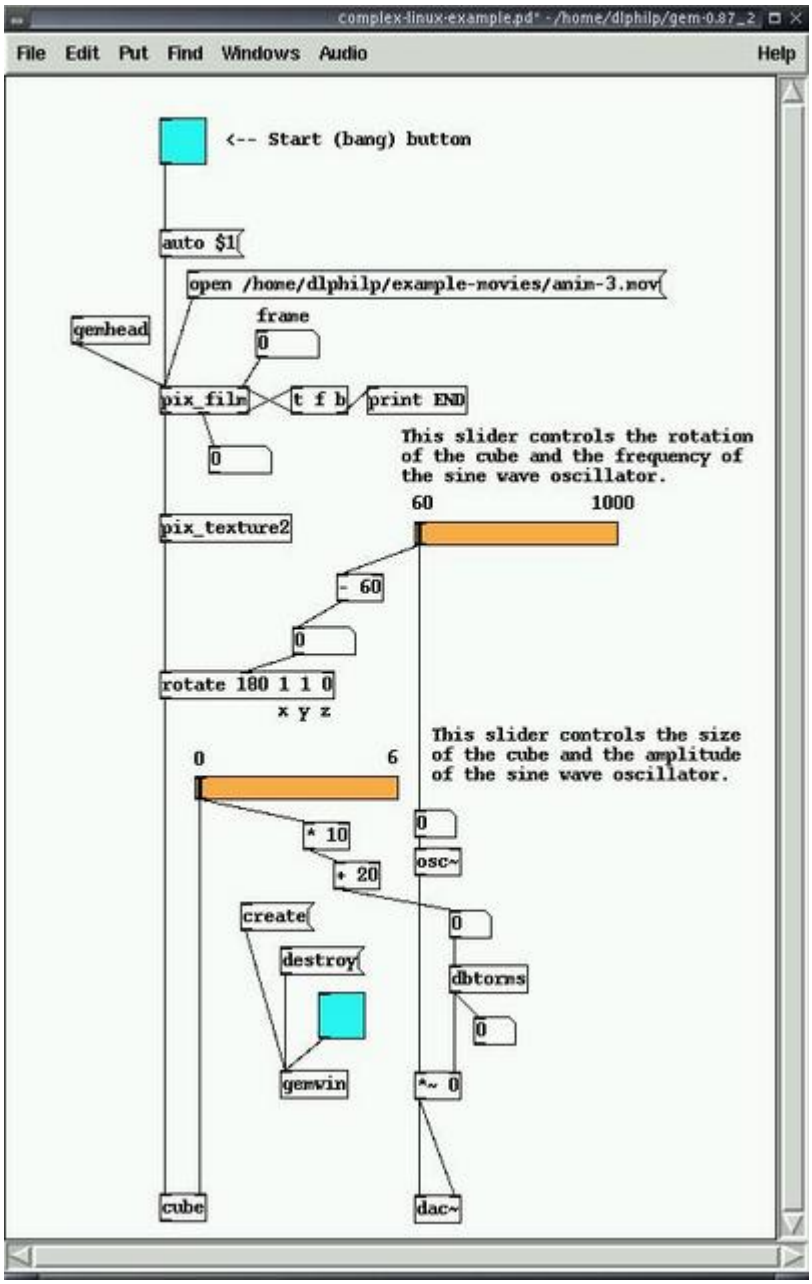


Figure 6. A Complex Pd/GEM Patch

Figure 6 illustrates our complex combined audio/video patch. As you can see, the two sliders each simultaneously control an aspect of the video along with an aspect of the audio. Adjusting the audio amplitude results in an adjustment

of the cube size, while moving the slider for the audio frequency control also will rotate the cube on its x/y axes. Multimedia artists will find great possibilities in Pd's support for such arbitrary attachments and correspondences. I also should note that GEM includes numerous other fascinating pixel-based effects (such as color convolution and pixel threshold control), but I leave their exploration to the interested reader.

Project Assessment

Basing our work upon the examples shown here, we are currently planning our presentation for a two-man show. We hope to use no more than two Linux-powered laptops (ideally we would need only one) and a variety of external devices (video recorder/player, still-image projector, lighting displays, etc.). Ease of transportation is a concern because we would like to be able to take the show on the road in a single vehicle.

Our Pd audio explorations have been quite stable in performance, which is good news because we plan to use Pd's real-time audio processing throughout the piece. GEM 0.87_2 sometimes crashes Pd, but I'm using a beta version as I write this article. IOhannes Zmoelnig is dedicated to GEM's improvement, so we reasonably can expect flawless performance by the time we are ready to mount our first performance (targeted for late 2002). We also have seen that combining heavy audio and video processing creates a need for more powerful hardware than we currently employ. I hope to improve our video situation with the addition of a GeForce3 video card. Finally, we also intend to use Pd's shoutcast~/oggcast~ objects for broadcasting our performances live over the Internet.

Conclusion

Pd is incredibly easy to work with, permitting fast construction of relatively complicated patches. The examples shown here were designed merely as learning tools and proof-of-concept demonstrations, and I have already created considerably more complex patches for our project. This article is only a shallow indicator of the possibilities of Pd and GEM, and I encourage all Linux-based audio and video artists to get involved with this software.

Acknowledgements

Vast thanks to Miller Puckette and Mark Danks for creating and freely distributing Pd and GEM. Thanks also to the Pd community for their continuing assistance to this perpetual newbie, and especially to IOhannes Zmoelnig of IEM for his beta version of GEM 0.87 and for his unstinting aid while I learn more about GEM's video objects. Finally, great thanks also must go to Guenter

Geiger for his initial work on the pix_movie object and for his many other contributions to the Linux versions of Pd and GEM.

Resources

Dave Phillips is a musician, teacher and writer living in Findlay, Ohio. He has been an active member of the Linux audio community since his first contact with Linux in 1995. He is the author of The Book of Linux Music & Sound, as well as numerous articles for *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Free Software in Brazil

Jon Hall

Issue #101, September 2002

More than saving money, the Software Livre movement offers Brazilian states control over their technological destinies.

Rio Grande do Sul (RS), one of 16 states in the country of Brazil, has been using free software (which they call *software livre*) for several years as a method of meeting their needs for production software in various areas. By using software livre, RS has been able to cut software costs. The money they do spend on software stays in the hands of Brazilian programmers, who buy Brazilian food, live in Brazilian houses and pay Brazilian taxes. None of the last three points is lost on the political leaders of RS, and these points are being learned not only by the other states in Brazil, but a lot of other countries in South America.

One successful project is called SAGU (sagu.codigoaberto.org.br), a unified open management system used to automate all of the relationships that exist between the academic, financial and administrative aspects of a grade school or university. Highly modular, everything from the entrance exam process to the financial accounting of overdue library book fees is handled by SAGU. As a flexible transaction-oriented database system, it allows for many other functions to be integrated. It can even be integrated with an external accounting and payroll system. Because a large part of teaching these days seems to be generating reports for local, state and federal governments, SAGU has its own reporting tool that generates PostScript documents on the fly. Originally conceived and implemented at Univates, a branch of the central university, it is now spreading throughout the Brazilian school and university system. And (of course) the whole project is software livre, so other school systems in other countries can benefit.

UERGS (Universidade Estadual do Rio Grande do Sul, www.uergs.rs.gov.br) is a project in which software is oriented toward distance learning. Designed for a

distributed university with more than 22 campuses, the UERGS Project helps manage the processes involved with that type of learning.

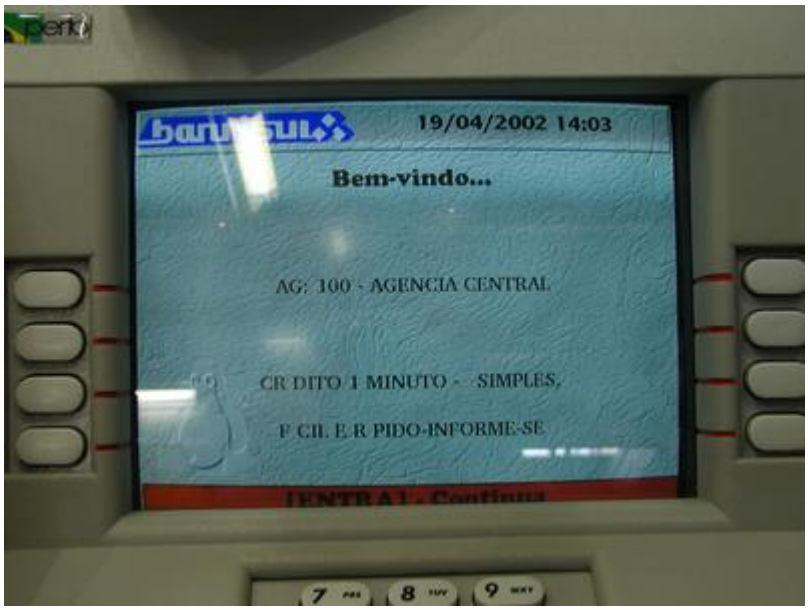
Directo GNU (www.direto.org.br), a corporate mailing solution based on software livre, is also underway. By using this e-mailing solution on the state's 60,000 computers, the state of RS will save over $10 million US. Through the use of local knowledge and expertise in setting it up, they also will guarantee the technological control of the software by RS far into the future. This is an important consideration if you are thinking about buying mission-critical software from outside your country.

Speaking of technological control, I was contacted by a company studying biodiversity in the Brazilian rain forest. Unfortunately, the software the company (located in Rio de Janeiro) needed was produced only in the United States, was prohibitively expensive and was also available only in English. This last point meant that they had to teach English to everyone who needed to use the software, at least as much English as was needed to use the software, read the manuals and so on. By using software livre, they were able to reduce the cost of the software, have it respond in Portuguese, tailor it exactly to their needs and keep Brazilian programmers and scientists paid. Thus, software livre allows a company that otherwise could not exist in Brazil to thrive and do good work.

Likewise, a project called Rede Escolar Livre RS (www.redeescolarlivre.rs.gov.br) will be supplying educational software to more than 2,000 of Brazil's public schools. The savings in individual productivity tools and network operational systems is estimated to be over $24 million US.

In the past few years I have noticed how important geographical information systems (GIS) are to almost everyone in the world. Most of you are familiar with mapping programs like MapQuest, MapBlast and others, but the use of GIS software in planning government and commercial actions is crucial (for more on this, please see www.freegis.org). In recognition of the need for GIS software to be free, the project Geoprocessamento Livre (notice the word “Livre” showing up more and more in these projects?) is a free GIS project to provide inexpensive GIS software. This GIS software is being used in many strategic projects, including those dealing with health and education.

The largest bank in RS, if not in all of Brazil, is using Linux in their new ATM machines. Manufactured by PERTO S.A., the TPC-2100 has the outline of a distinctive penguin on the CRT screen before you start your transactions. Go Tux!



Distinctive Penguin on ATMs in Brazil

Probably the most humanly touching project is headed up by a person named Pascal and concerns the Landless Rural Workers Movement of Brazil (www.mstbrazil.org). Brazil was settled by a few very wealthy families who continue to control the vast majority of land in Brazil today. This land ownership tends to impoverish many in Brazil by withholding the right to grow their own food or to otherwise earn a living. A movement to help these landless workers has formed in Brazil, and they have sent a couple dozen people trained in proprietary software to a central location in RS to be retrained as Linux system administrators. One woman had traveled four days by bus to get to the training and would now have to travel back the same way. It was an intensive month-long course that made each student a part of the fellowship of software livre and the Landless Rural Workers Movement.


maddog with the Linux System Administrator Class for the Landless Rural Workers Movement

As I travel to various countries that I view to be emerging economies, I see governments realizing that software livre means more than cheap software. It also means software tailored to their non-English consumers. It means jobs for their local programmers. It means industries that can happen inside their own countries, due to the availability of software that allows these industries to

flourish. It means control over their own destiny. It literally means software livre.

email: maddog@li.org

Jon “maddog” Hall is founder and executive director of Linux International.

Archive Index Issue Table of Contents

Advanced search

2002 Editors' Choice Awards

LJ Staff

Issue #101, September 2002

We present this year's winners and a few honorable mentions.

This year, in order to draw upon a wider base of knowledge and experience, we made some modifications to the process of choosing the winners of the *Linux Journal* Editors' Choice Awards. We started by selecting a board of over 50 Linux experts, chosen largely from among the best and brightest of *Linux Journal* contributors. This board, after receiving the categories, was charged with coming up with nominees for each. Once we had the nominees and the board's comments on each one, we passed them on to our contributing editors for their input. Armed with this information, our editorial team made the final decisions.

This year, we see an unusual number of free and open-source software products among the winners. This is not a sign that commercial products are in decline, either in quantity or quality, but rather a reflection of the maturity of many open-source projects. There were a number of commercial products that also ranked highly with the nomination board, and in those cases where one product received a high number of nominations, but won no award, we included it as an honorable mention.

Server Appliance: SnapGear Lite/Lite+ SOHO Firewall/VPN Client

We ran a review of the SnapGear Lite in the *LJ* April 2002 issue (www.linuxjournal.com/article/5744) and concluded that for the price and functionality, it can't be beat. For $249-$299 US (depending on whether you get the Lite or the Lite+) you get, in addition to firewall and NAT functionality, a hardware-based VPN client that can emulate Microsoft's Point-to-Point Tunneling Protocol, letting you use a Linux machine to access a Microsoft VPN.

SnapGear's products are based on the SecureEdge platform, developed when SnapGear was merged with Lineo (in the days when Lineo was in the business

of acquiring companies). SnapGear has spun off into its own company again for some time now and seems to be doing very well with their line of Linux-based routers, which allows them pricing significantly lower than much of their competition.

Honorable Mention: Sun Microsystems' Cobalt Qube

Security Tool: GPG

Did you notice how much more cryptographically signed e-mail you got during the past year? You should thank the developers of your favorite mailer for making mail signing, encryption and checking easy, but most of all, thank the GNU Privacy Guard developers for offering a compatible replacement for the original Pretty Good Privacy, which vanished in a flurry of—all together now—*Corporate Shenanigans*. Now, there's no excuse for not being able to send and receive secure mail.

Web Server: IBM X-Series

It seems like everybody's making rackmount Linux web servers. What's IBM have that the others don't? A smooth web ordering process, whatever service and support level you desire, and they'll support their hardware running Red Hat, Caldera, SuSE or Turbolinux. Quite a choice.

Honorable Mention: Sun Microsystems' Cobalt RaQ XTR

Enterprise Application Server: Zope

Before the Web, how often did DBAs and graphic designers get a chance to call each other productivity-sucking idiots or worse? Ever since web site management got big and professional, we've known that graphic designers don't want to work on templates, but sites full of static luscious-looking pages get unmanageable real fast, and the answer is a database and a templating system.

Now, let the graphics people work with WYSIWYG tools if they like—Zope offers a clever templating system that makes the templates work in the WYSIWYG tool when it's time to modify them. Everyone else will appreciate the load-balancing capability and, of course, the Free Software license.

Technical Workstation: HP x4000

We first saw this machine at LinuxWorld New York, 2002 where it was being displayed with high-end graphics applications such as Maya for Linux. Our impressions of its high-performance capabilities were confirmed by the review

we ran in the *LJ* June 2002 issue, in which reviewer Thad Beier used the words “shockingly faster” to describe the x4000 in comparison with machines he was used to. Thad used the machine to run resource-gobbling effects software and simply was blown away with the performance. As tested, with two 2.2GHz processors, 4GB of RDRAM, it's not hard to imagine that he would be. Of course HP ships the x4000 with Linux (Red Hat) preloaded. HP offers the x4000 in a number of configuration options, so you can get what you need regardless of whether you're running an effects studio or running complex Verilog simulations with Icarus.

Web Client: Mozilla and Galeon

As some web browsers have grown huge with features and others have gone the lean and fast route, we chose two winners. So ask yourself: do you like your web browser thick and juicy or simply as a thin component of your desktop? Either way, we don't cut the browser any slack when it comes to honoring the W3C's standards. Web standards are the only reason we can use the software of our choice to browse sites that webmasters create with the software of their choice—it's the social contract that underlies freedom. “No browser does a better job of standards compliance” is what the Web Standards Project says about Mozilla, and that's good for everyone. So pick Mozilla, the super-deluxe, super-themeable browser, and get mail, news, password management and other power features, or get Galeon, a light browser that doesn't duplicate your other GNOME applications.

Honorable Mention: Konqueror

Graphics Application: The GIMP

If you're like most Linux users, you fire up The GIMP for miscellaneous image tasks such as converting and cropping photos for your web site. But The GIMP is much more than that. It's becoming one of those great platforms, like Perl and Apache, that becomes a natural starting point for a development and support community. The GIMP has a lot of functionality that takes awhile to learn, including not one but two built-in scripting languages. Check out manual.gimp.org for an on-line manual.

Communication Tool: Evolution Mailer

We've been watching our contributors' headers to see what mailers they use, and the unthinkable is happening. Linux gurus are dropping text-based mailers for a GUI mailer called Evolution (more on this disturbing situation as it develops). Besides mail, Evolution also offers a calendar and to-do list. We like the idea of being able to compose more than one message at once, but our vi-

trained fingers wouldn't get very far without integrating Jason Hildebrand's gnome-vim.

Consumer Software: KDE 3.0

KDE 2.0 represented real progress toward making Linux a more viable option for many people on the desktop. There was a lot of great ideas, such as a super-customizable desktop. Unfortunately, there was also plenty that didn't work, or at least didn't work right. KDE 3.0, while it doesn't look much different, offers the goods that KDE 2.0 seemed to promise and more, including a much higher level of stability and new functionality for many applications, including Konqueror (like the ability to disable JavaScript pop-up windows) and KMail. KDE 3.0 gives all these GUI goodies without forgetting the command-line user: Konsole, the KDE terminal window, also comes with additional functionality and lets you monitor for new (or no) activity.

Development Tool: Emacs

With all the impressive development tools for Linux coming out of late, it's easy to ignore the extensive IDE capabilities of Emacs, as Charles Curley points out in his article on Emacs in the *LJ* June 2002 issue. Emacs' high level of support for customization makes it a favorite among hackers. Not only does it support many languages, but features such as Electric C (for automation of indentation and pretty printing), spell checking and the ability to act as a front end for GCC, GDB and CVS make it a sensible choice for a lot of programming needs. For those unaccustomed to the Free Software world, it's hard to believe it's free—and it's been there all along.

Honorable Mentions: KDevelop and Borland's Kylix

Database: MySQL

If you're one of the people who has been saying, “I can't use MySQL because it doesn't have [feature you need here]”, it's time to read up on MySQL 4.0 and try it out on a development system. Can you say, “full support for transactions and row-level locking”? “UNION”? “Full text search”?

The new MySQL is even available as a library you can compile into your application. Proprietary licenses are available if you can't use the GPL.

Honorable Mention: PostgreSQL

Backup Software: Sistina Software's Logical Volume Manager for Linux

No matter what your backup plan is, and what hardware and software you use to handle the mundane details of copying your working files to off-line storage, you need to make a copy that's internally consistent. This is especially critical when you're backing up a database. (For a simple example, say that you keep your users' home directories by state, and Joe moves from /home/washington/joe to /home/alabama/joe while you're backing up missouri. Where's Joe's home directory on the tape? Nowhere!)

Expensive proprietary UNIX systems have had a solution for years: filesystems that support taking a “snapshot”, which looks like your working filesystem frozen in time. Instead of “shut down the database, dump it to tape, start up the database”, it's “shut down the database, snapshot, start up the database, dump the snapshot to tape”—quite a time-saver. Thanks to Sistina Software, Linux now has this essential feature for backing up busy servers.

Office Application: OpenOffice 1.0

With Sun now charging for StarOffice 6.0 (with its increased functionality and the proprietary elements that go with it), it's very nice that they support OpenOffice and continue to make it available for free. For those who can do without certain features such as document templates and a grammar checker, OpenOffice offers an amazing amount of functionality and advanced features that come very close indeed to matching those of MS Word. Some of these features are autocorrect/autoformat modes, the ability to compare documents and include cross-references, fields, an equation editor and global customization settings. OpenOffice can import OLE objects and charts and does a pretty good job of both importing and exporting files in the MS Word format.

Mobile Device: Sharp Zaurus

See “Product of the Year” below.

Training and Certification Program: LPI

The Linux Professional Institute reports that more than 10,000 people have taken the exams to become LPI-certified. Training is available in classroom settings or on IBM's developerWorks web site, and the exams are tough but fair. Certification can't promise you a job, but if you get a chance to go for it, it shows potential employers you are keeping your skills up to date.

Game: *TuxRacer*

This little open-source game has had more than one million downloads. It offers nice 3-D graphics, and you even can create your own courses with almost

any paint program (The GIMP, for instance). As former contributing editor Neil Doane says, “If there's anything better than 3-D high-speed belly surfing downhill competition penguin racing, I haven't found it.”

Honorable Mention: *PySol*

Books: *Linux Device Drivers, 2nd Edition* by Alessandro Rubini and Jonathan Corbet and *The Future of Ideas: The Fate of the Commons in a Connected World* by Lawrence Lessig

With the number of political and legal issues surrounding free and open-source software, and the books written that address them, we thought it best to choose a winner for both a technical and a nontechnical book this year.

Linux Device Drivers has been a must-have book for people getting into kernel development, and 2001 saw the release of a 2.4-oriented second edition. You'll still need your *Linux Journal* subscription for some of the newer stuff, but this book is an excellent introduction and field guide to the source code. The paper format is handy, but you can use the on-line version for previewing and as a quick reference.

In *The Future of Ideas: The Fate of the Commons in a Connected World*, Lawrence Lessig writes,

> A time is marked not so much by ideas that are argued about as by ideas that are taken for granted. The character of an era hangs upon what needs no defense. Power runs with ideas that only the crazy would draw into doubt. The “taken for granted” is the test of sanity; “what everyone knows” is the line between us and them.

Thanks to Larry and his latest book, the public nature of the Net has a far better chance of being taken for granted than it ever would have had without them.

Before Larry began writing and talking about the “end-to-end” architecture of the Net and its place-like nature as a “Commons”, those ideas were taken for granted by a rather small *us*—a population that surely included the majority of *Linux Journal* readers. It may take some time before everyone knows and agrees with these ideas; but they are spreading fast. After they achieve ubiquity (and we have faith they will), Professor Lessig will be remembered as one of the Net's true heroes.

Web Site: Google

What more can you say about the world's growing dependency on more than 10,000 Linux boxes behind the most popular search engine? Never in the

history of the Web has there been a site that has done more with less hype than Google. Its few self-serving messages do little or nothing to compromise the vast white space that surrounds the one thing people come there to use: the search box. Paid advertising appears alongside search results, but it never intrudes. And so much of it is useful to both seller and searcher that Google actually has a going business that makes money.

In the last year or so the company has added image and newsgroup searches to its front page, and a catalog engine has been in the works as well. And lately, the company has added an API that lets developers query web documents using SOAP and WSDL protocols for noncommercial purposes.

The search engine also acts, in an oblique manner, as an anti-censorship tool. Google will remove pages from its index in response to take-down letters written under the controversial Digital Millennium Copyright Act, but there's a catch for would-be DMCA censors. After the Church of Scientology attacked the exposé site xenu.net, Google began reporting take downs to the free speech watchdog site chillingeffects.org. When corporations try to censor Google results, they just bring more attention to their victims.

While Google's policies may not please everybody, it has kept better faith with users than any other search engine, and for that it deserves all its ample success.

Product of the Year: Sharp Zaurus

Two words describe this past year: Zaurus Frenzy. We dig Trolltech's palmtop Qtopia environment, based on the popular Qt toolkit. It's got all the classic PDA stuff, such as address book and calendar, plus a selection of nifty games and good office and web software.

The Qtopia development boom owes a lot to the shining promise of the first Qtopia-based product, Sharp's Zaurus PDA, which offers all the hardware features you might want, including optional wireless networking, high-quality audio, a keyboard and a replaceable, rechargeable battery.

Our embedded Linux news archive for 2001 was full of companies committing to offer applications, services and training for Qtopia and the Zaurus. When you get one you'll know why.

Archive Index Issue Table of Contents

Advanced search

Fire, Brimstone and Real-Time Linux

Rick Lehrbaum

Issue #101, September 2002

Debate continues over the best approach to real-time capabilities and the Linux kernel.

Though fewer than 10% (or less) of embedded Linux applications actually require real-time enhancements or add-ons, articles and discussions on that subject invariably spark passionate debate. For whatever reason, the topic is a magnet for what might be characterized best as a sort of religious fervor. So it is with some tiptoeing through a minefield that this month's embedded column begins with several topics related to real-time Linux. Hold on to your hats.

VDC Sees Real-Time Linux Support Opportunity

In their recently published research report “Linux's Future in the Embedded Systems Market”, Venture Development Corp. (VDC) concluded that the availability of real-time solutions for Linux are needed to accelerate the broad adoption of Linux in embedded designs.

The report analyzes the current size and future growth of the worldwide market for embedded Linux software solutions (www.vdc-corp.com). According to the study, the dominant factors favoring the use of Linux in embedded projects include the availability of source code, royalty-free options and reliability. On the other hand, VDC found “real-time limitations” to be the most common issue cited by embedded developers as inhibiting their adoption of Linux for future projects.

Here is a list, in ranked order, of what developers told VDC were their main concerns with respect to using Linux in embedded systems and devices:

1. Real-time limitations
2. Doubts about availability and quality of support
3. Fragmentation concerns

4. Doubts about vendor longevity
5. Footprint size
6. Other

The Great Real-Time Linux Debate (Redux)

The usual real-time debate erupted shortly after *Embedded Linux Journal* published the third article in a series by Kevin Dankwardt on real-time Linux technologies. Here's an outline of the sequence of reactions and responses from key players in the real-time Linux market, followed by a pointer to where you can read them all on-line:

- MontaVista Software's Kevin Morgan issued a response to Dankwardt's article in which he offered “a few clarifications (or points of view)”.
- Victor Yodaiken and Matt Sherer (of FSMLabs) reacted to Kevin Morgan's response to Dankwardt's article, taking exception to Morgan's assertion that RTLinux is “not appropriate for the placement of comprehensive applications”.
- Kevin Morgan clarified the status of MontaVista's kernel preemption enhancements and responded to several other issues raised in Yodaiken and Sherer's earlier comments.
- Karim Yaghmour provided “the RTAI perspective”--drawing attention to the nature of the API, the usability of the methods and distinctions in the overall openness of the specific approaches being compared.
- Doug Locke, TimeSys' VP of technology, contrasted his company's preemptible Linux implementation with the one pioneered by MontaVista and commented on several aspects of the preceding debate.

You can access all the above, including the original three-part *ELJ* article by Kevin Dankwardt, from this summary page: www.linuxdevices.com/news/NS4265889552.html.

Still More on Real Time

Clark Williams of Red Hat wrote a whitepaper titled “Linux Scheduler Latency”, in which he compares the performance of two popular methods for improving Linux kernel preemption latency—the preemption patch pioneered by MontaVista and the low-latency patch pioneered by Ingo Molnar—and discovers that the best approach might be a combination of the two (www.linuxdevices.com/articles/AT8906594941.html).

The ADEOS Project announced its first release of ADEOS, a hardware abstraction layer that allows a real-time kernel and a general-purpose kernel to coexist on one CPU. The announcement claims that “RTAI will eventually use

ADEOS services, thus offering a real-time kernel based on a principle clearly different from the 5,995,745 US Patent", aka the "RTLinux patent" (www.freesoftware.fsf.org/adeos).

Victor Yodaiken published a whitepaper that points out the disadvantages of dealing with the issue of "priority inversion" in real-time systems by means of a commonly used method known as "priority inheritance". Priority inversion refers to the situation when a scheduled task must wait for a lower-priority task to complete. In the whitepaper, Yodaiken describes "the classical nightmare case" of priority inversion as being "when a low priority task owns a resource, a high priority task is blocked [and] waiting for the resource, and intermediate priority tasks keep preempting the low priority task so it cannot make progress toward releasing the resource." Yodaiken says the technique of priority inheritance is intended to allow:

> a task that is blocked waiting for a resource [to pass] its priority down to the owner. The low priority task is [thus] considered to be acting on behalf of the highest priority blocked task and inheritance prevents intermediate priority tasks from interfering.

However, "priority inheritance is neither efficient nor reliable", the paper argues, and its "implementations are either incomplete (and unreliable) or surprisingly complex and intrusive", asserts Yodaiken (www.linuxdevices.com/articles/AT7168794919.html).

Red Hat "Adjusts" Its Embedded Strategy

Red Hat discontinued development of its eCos open-source embedded operating system and is rumored to have discontinued many of its embedded Linux development efforts.

Asked whether Red Hat was still in the embedded market, Red Hat Chief Technology Officer Michael Tiemann replied, "Yes—but our strategy is to expand the scope of Linux to encompass the embedded space." Expanding on this point, Tiemann said "the embedded world that [Red Hat is] most interested in needs a Linux platform that extends into the embedded space", as opposed to a unique version of Linux tailored specifically to embedded devices.

These statements explain Red Hat's move away from products like eCos, an open-source embedded operating system that Red Hat inherited via its acquisition of Cygnus Solutions in late 1999, and µClinux, a version of Linux geared toward resource-constrained, "deeply embedded" devices that Red Hat got involved in via its mid-2000 acquisition of Wirespeed. In addition, Tiemann's reduced investment statements explain Red Hat's developing embedded-oriented tools, like the Embedded Linux Developers Suite (ELDS).

Tiemann elaborates on what he means by "a Linux platform that extends into the embedded space" in a guest editorial for LinuxDevices.com entitled, "How Linux will Revolutionize the Embedded Market" (www.linuxdevices.com/articles/AT7248149889.html). That editorial basically unfolds a strategy that treats the embedded market as a portion of a continuum—one that increasingly demands greater consistency of technology, APIs, middleware and tools, due to growing end-to-end application connectivity and interoperability.

The best approach, Tiemann argues, is to offer a solution that meets the needs of the entire range of requirements within a single *platform*, rather than providing a unique version of Linux specially tailored to embedded systems. Quoting from the conclusion,

> The deeper I look into environments adopting Linux, from embedded to enterprise, the more I believe that Linux has the requisite DNA and development model to scale truly from embedded to enterprise as a single platform, and Red Hat's focus will remain on ensuring that what works for the mainframe, and the server, and the workstation, also works for the appliance, the carrier, the router, the PDA, and the cell phone; and, of course, vice-versa.

Three Reviews on Hollabaugh's *Embedded Linux*

Finally, here are links to three excellent on-line reviews of Craig Hollabaugh's well-received book, *Embedded Linux*:

- Two reviews from the Embedded Linux Consortium, by Joel Williams and Dr. Ian McLoughlin (FTP download available at embedded-linux.org/files/Review1.pdf).
- One from LinuxDevices.com, by Jerry Epplin (www.linuxdevices.com/articles/AT8515229385.html).

Embedded Linux is published by Addison-Wesley Professional (ISBN: 0672322269) and is available at various on-line retailers. Amazon.com provides 43 sample pages of the book on their web site.

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.

Archive Index Issue Table of Contents

Advanced search

Memory Leak Detection in Embedded Systems

Cal Erickson

Issue #101, September 2002

Cal discusses mtrace, dmalloc and memwatch—three easy-to-use tools that find most application program errors.

One of the problems with developing embedded systems is the detection of memory leaks; I've found three tools that are useful for this. These tools are used to detect application program errors, not kernel memory leaks. Two of these tools (mtrace and dmalloc) are part of the MontaVista Linux Professional Edition 2.1 product. The other (memwatch) is available from the Web (see Resources).

C and C++ programmers control dynamic memory allocation. Reckless use of this control can lead to memory management problems, which cause performance degradation, unpredictable execution or crashes.

Some of the problems that cause memory leaks are writing or reading beyond an allocated memory segment or trying to free memory that has already been freed. A memory leak occurs when memory is allocated and not freed after use, or when the pointer to a memory allocation is deleted, rendering the memory no longer usable. Memory leaks degrade performance due to increased paging, and over time, cause a program to run out of memory and crash. Access errors lead to data corruption, which causes a program to behave incorrectly or crash. When a program runs out of memory it also can cause the Linux kernel to crash.

Designing and programming an embedded application requires great care. The application must be robust enough to handle every possible error that can occur; care should be taken to anticipate these errors and handle them accordingly—especially in the area of memory. Often an application can run for some time before it mysteriously crashes itself or the system as a result of a memory allocation that is never freed. Finding these errors can be done through use of memory leak detectors.

These tools work by replacing malloc, free and other memory management calls. Each tool has code that intercepts calls to malloc (and other functions) and sets up tracking information for each memory request. Some tools implement memory protection fences to catch errant memory accesses.

Some of the leak detection programs are very large and require a virtual memory image of the program being searched. This requirement makes it very difficult to use on embedded systems. However, mtrace, memwatch and dmalloc are simple programs that find most errors.

All three tools were run on one example C program containing common memory handling errors. This program, together with Makefiles for building it with the three tools, is available as a downloadable file at ftp.linuxjournal.com/pub/lj/listings/issue101/6059.tgz. All of these tools have been used in several different target architectures. The example code will work whether compiled natively or cross-compiled.

mtrace

The simplest of the three tools is mtrace. A feature of the GNU C library, mtrace allows detection of memory leaks caused by unbalanced malloc/free calls. It is implemented as a function call, mtrace(), which turns on tracing and creates a log file of addresses malloc'd and freed. A Perl script, also called mtrace, displays the log file, listing only the unbalanced combinations and—if the source file is available—the line number of the source where the malloc occurred. The tool can be used to check both C and C++ programs under Linux. One of the features that makes mtrace desirable is the fact that it is scalable. It can be used to do overall program debugging but can be scaled to work on a module basis as well.

Key to the use of the mtrace feature are three items: include mcheck.h, set the MALLOC_TRACE environment variable and call the mtrace() function call. If the MALLOC_TRACE variable is not set, mtrace() does nothing.

The output of mtrace includes messages such as:

```
- 0x0804a0f8 Free 13 was never alloc'd
/memory_leak/memory_leaks/mtrace/my_test.c:193
```

to indicate memory that was freed but never malloc'd and a “Memory not freed” section that includes the address, size and line number of calls to malloc for which no free occurred.

memwatch

memwatch is a program that not only detects malloc and free errors but also fencepost conditions. Fencepost conditions occur when writing data into an allocated chunk of memory (allocated by malloc) and the data goes beyond the end of the allocated area. Some things that memwatch does not catch are writing to an address that has been freed and reading data from outside the allocated memory.

The heart of memwatch is the memwatch.c file. It implements the wrappers and code for the address checking. To use memwatch the file memwatch.h must be included in the source. The variables MEMWATCH and MW_STDIO must be defined on the compile command line (-DMEMWATCH and -DMW_STDIO). The memwatch.c file must be used with the application as well. The object module from the compile of memwatch.c must be included in the link of the application. Upon execution of the application, a message will appear on stdout if memwatch found any abnormalities. The file memwatch.log is created that contains the information about the errors encountered. Each error message contains the line number and source-code filename where the error occurred.

Comparing memwatch.log with the log from mtrace, the same errors are reported. The memwatch tool also found a fencepost condition where the memory addresses were changed to overwrite the start and end of an allocated area, showing the expanded capability of memwatch in this case. The disadvantage is that memwatch is not scalable. It has to run on the whole application.

dmalloc

The third tool is a library that is designed as a drop-in substitute for malloc, realloc, calloc, free and other memory management functions. It provides runtime configurability. The features of the tool provide memory leak tracing and fencepost write detection. It reports its errors by filename and line number and logs some general statistics. This library, created and maintained by Gray Watson, has been ported to many operating systems other than Linux.

The package is configurable to include thread support and C++ support. It can be built both as shared and static libraries. All of these options are selected when building the tool. The result is a set of libraries that are used when linking the application program. There is an include file (dmalloc.h) that needs to be included in the source of the application to be checked. In addition to the library and include file, it is necessary to have an environment variable set up that dmalloc reads to configure how it checks and where it puts the logging

information. The following line is the setup used with the test program for dmalloc:

```
export \
DMALLOC_OPTIONS=debug=0x44a40503,inter=1,log=logfile
```

What this means is 1) log is a file named logfile in the current directory, 2) inter is the frequency for the library to check itself and 3) debug is a hex number whose bits select the types of checking to do. This example tests for just about every possible error. The following is a list of the tests and the corresponding bits to set in “debug”:

- none (nil): no functionality (0)
- log-stats (lst): log general statistics (0x1)
- log-non-free (lnf): log non-freed pointers (0x2)
- log-known (lkn): log only known non-freed (0x4)
- log-trans (ltr): log memory transactions (0x8)
- log-admin (lad): log administrative info (0x20)
- log-blocks (lbl): log blocks when heap-map (0x40)
- log-bad-space (lbs): dump space from bad pointers (0x100)
- log-nonfree-space (lns): dump space from non-freed pointers (0x200)
- log-elapsed-time (let): log elapsed time for allocated pointer (0x40000)
- log-current-time (lct): log current time for allocated pointer (0x80000)
- check-fence (cfe): check fencepost errors (0x400)
- check-heap (che): check heap adm structs (0x800)
- check-lists (cli): check free lists (0x1000)
- check-blank (cbl): check mem overwritten by alloc-blank, free-blank (0x2000)
- check-funcs (cfu): check functions (0x4000)
- force-linear (fli): force heap-space to be linear (0x10000)
- catch-signals (csi): shut down program on SIGHUP, SIGINT, SIGTERM (0x20000)
- realloc-copy (rco): copy all re-allocations (0x100000)
- free-blank (fbl): overwrite freed memory space with BLANK_CHAR (0x200000)
- error-abort (eab): abort immediately on error (0x400000)
- alloc-blank (abl): overwrite newly alloced memory with BLANK_CHAR (0x800000)
- heap-check-map (hcm): log heap-map on heap-check (0x1000000)
- print-messages (pme): write messages to stderr (0x2000000)
- catch-null (cnu): abort if no memory available (0x4000000)

- never-reuse (nre): never reuse freed memory (0x8000000)
- allow-free-null (afn): allow the frees of NULL pointers (0x20000000)
- error-dump (edu): dump core on error and then continue (0x40000000)

If the library needs to check C++ programs, a source file named dmalloc.cc is needed with the application. This module provides wrapper functions for new to malloc and delete to free. The GNU debugger GDB can be used with dmalloc. A file is included with the product that can be used as part of a .gdbinit file so that GDB is set up automatically to know about dmalloc.

Along with the library is a utility named dmalloc that will programmatically set up the DMALLOC_OPTIONS variable. I've made a setup script that is sourced prior to running the program to be debugged. This way the setup is repeatable without errors.

This article only covers the general use of the tool, but the documentation for it is extensive and details more available features. The test program used with the earlier tools was run using DMALLOC. The results can be lengthy and optionally include the bytes present at critical areas of memory such as the “fence-top”, where a pointer overruns a malloc'd area. The end of the log file includes statistics, addresses, block sizes and line numbers for occurrences of malloc without free.

The three tools provide varying support for memory leak detection and reporting. Each of these tools has been used on a Linux workstation as well as cross-compiled and executed on several different target architectures. In one application, developers used all three tools. **mtrace** was used to find a memory leak in a third-party C++ library where an exception throw/catch block caused a major leak. The dmalloc tool was used to find memory leaks in the execution of Linux pthreaded applications. The memwatch tool was used to catch a buffer pool mechanism that was not properly defragmenting itself. These tools are small and easy to implement and remove when debugging is completed.

The example program consists of one source file, my_test.c. There are three separate directories that contain a README, Makefile and a script to run the test [available at ftp.linuxjournal.com/pub/lj/listings/issue101/6059.tgz]. In the dmalloc test, the environment setup script is also available. Each of the tests has been built on Red Hat and SuSE Linux releases, as well as the MontaVista Linux cross-development environments.

Resources

Cal Erickson (cal_erickson@mvista.com) currently works for MontaVista Software as a senior Linux consultant. Prior to joining MontaVista, he was a senior support engineer at Mentor Graphics Embedded Software Division. Cal has been in the computing industry for over 30 years with experience at computer manufacturers and end-user development environments.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

In-Memory Database Systems

Steve Graves

Issue #101, September 2002

IMDSes are especially useful for embedded development, where every saved process shrinks the footprint and the bottom line.

Growth in intelligent connected devices is soaring. Whether in the home, the pocket or built into industrial communications and transportation systems, such gear has evolved to include powerful CPUs and sophisticated embedded systems software. One type of software increasingly seen within such devices is the database management system (DBMS). While familiar on desktops and servers, databases are a recent arrival to embedded systems. Like any organism dropped into a new environment, databases must evolve. A new type of DBMS, the in-memory database system (IMDS), represents the latest step in DBMSes' adaptation to embedded systems.

Why are embedded systems developers turning to databases? Market competition requires that devices like set-top boxes, network switches and consumer electronics become “smarter”. To support expanding feature sets, applications generally must manage larger volumes of more complex data. As a result, many device developers find they are outgrowing self-developed data management solutions, which can be especially difficult to maintain and extend as application requirements increase.

In addition, the trend toward standard, commercial off-the-shelf (COTS) embedded operating systems—and away from a fragmented environment of many proprietary systems—promotes the availability of databases. The emergence of a widely used OS such as embedded Linux creates a user community, which in turn spurs development (both commercially and noncommercially) of databases and other tools to enhance the platform.

So device developers are turning to commercial databases, but existing embedded DBMS software has not provided the ideal fit. Embedded databases emerged well over a decade ago to support business systems, with features

including complex caching logic and abnormal termination recovery. But on a device, within a set-top box or next-generation fax machine, for example, these abilities are often unnecessary and cause the application to exceed available memory and CPU resources.

In addition, traditional databases are built to store data on disk. Disk I/O, as a mechanical process, is tremendously expensive in terms of performance. This often makes traditional databases too slow for embedded systems that require real-time performance.

In-memory databases have emerged specifically to meet the performance needs and resource availability in embedded systems. As the name implies, IMDSes reside entirely in memory—they never go to disk.

So is an IMDS simply a traditional database that's been loaded into memory? That's a fair question because disk I/O elimination is the best-known aspect of this new technology. The capability to create a RAM disk, a filesystem in memory, is built into Linux. Wouldn't deploying a well-known database system, such as MySQL or even Oracle, on such a disk provide the same benefits?

In fact, IMDSes are considerably different beasts from their embedded DBMS cousins. Compared to traditional databases, IMDSes are less complex. Beyond the elimination of disk I/O, in-memory database systems have fewer moving parts or interacting processes. This leads to greater frugality in RAM and CPU use and faster overall responsiveness than can be achieved by deploying a traditional DBMS in memory. An understanding of what's been designed out of, or significantly modified in, IMDSes is important in deciding whether such a technology suits a given project. Three key differences are described below.

Caching

Due to the performance drain caused by physical disk access, virtually all traditional DBMS software incorporates caching to keep the most recently used portions of the database in memory. Caching logic includes cache synchronization, which makes sure that an image of a database page in cache is consistent with the physical database page on disk. Cache lookup also is included, which determines if data requested by the application is in cache; if not, the page is retrieved and added to the cache for future reference.

These processes play out regardless of whether a disk-based DBMS is deployed in memory, such as on a RAM disk. By eliminating caching, IMDS databases remove a significant source of complexity and performance overhead, and in the process slim down the RAM and CPU requirements of the IMDS.

Data-Transfer Overhead

Consider the handoffs required for an application to read a piece of data from a traditional disk-based database, modify it and write that piece of data back to the database. The process is illustrated in Figure 1.

1. The application requests the data item from the database runtime through the database API.
2. The database runtime instructs the filesystem to retrieve the data from the physical media.
3. The filesystem makes a copy of the data for its cache and passes another copy to the database.
4. The database keeps one copy in its cache and passes another copy to the application.
5. The application modifies its copy and passes it back to the database through the database API.
6. The database runtime copies the modified data item back to database cache.
7. The copy in the database cache is eventually written to the filesystem, where it is updated in the filesystem cache.
8. Finally, the data is written back to the physical media.

These steps cannot be turned off in a traditional database, even when processing takes place entirely within memory. And this simplified scenario doesn't account for the additional copies and transfers required for transaction logging!

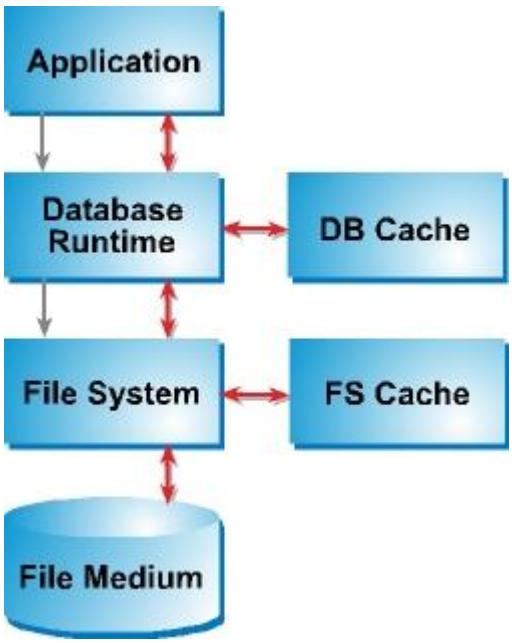


Figure 1. Data flow in a traditional DBMS. Red lines represent data transfer. Gray lines represent message path.

In contrast, an in-memory database system entails little or no data transfer. The application may make copies of the data in local program variables, but it is not required. Instead, the IMDS gives the application a pointer that refers directly to the data item in the database, enabling the application to work with the data directly. The data is still protected because the pointer is used only through the database API, which insures that it is used properly. Elimination of multiple data transfers streamlines processing. Cutting multiple data copies reduces memory consumption, and the simplicity of this design makes for greater reliability.

Transaction Processing

In the event of a catastrophic failure, such as loss of power, a disk-based database recovers by committing complete transactions or rolling back partial transactions from log files when the system restarts. Disk-based databases are hard-wired to keep transaction logs, to flush transaction log files and to cache to disk after transactions are committed.

Main memory databases also provide transactional integrity. To do this, the IMDS maintains a before image of the objects that are updated or deleted and a list of database pages added during a transaction. When the application commits the transaction, the memory for before images and page references returns to the memory pool (a fast and efficient process). If an in-memory database must abort a transaction (for example, if the inbound data stream is interrupted), the before images are restored to the database and the newly inserted pages are returned to the memory.

In the event of catastrophic failure, the in-memory database image is lost. This is a major difference from disk-based databases. If the system is turned off, the IMDS is reprovisioned upon restart. Consequently, there is no reason to keep transaction log files, and another complex, memory-intensive task is eliminated from the IMDS.

This functionality may not suit every application, but in the embedded systems arena, examples abound of applications with data stores that can be easily replenished in real time. These include a program guide application in a set-top box that downloads from a satellite or cable head-end, a wireless access point provisioned by a server upstream or an IP routing table that is repopulated as protocols discover network topology. Developers of such systems gladly limit the scope of transaction processing in exchange for superior performance and a smaller footprint.

This does not preclude the use of saved local data. With an IMDS, the application can open a stream (a socket, pipe or a file pointer) and instruct the database runtime to read or write a database image from or to the stream. This feature could be used to create and maintain boot-stage data, i.e., an initial

starting point for the database. The other end of the stream can be a pipe to another process or a filesystem pointer (any filesystem, whether it's magnetic, optical or Flash).

Application Scenario: IP Routers

Where and how can IMDS technology make a difference? While in-memory databases have cropped up in various application settings, the following scenario, involving embedded software in the most common internet infrastructure device—the IP router, offers an idea of the problems this technology can address.

Modern IP routers incorporate routing table management (RTM) software that accomplishes the core task of determining the next hop for data packets on the Internet and other networks. Routing protocols continuously monitor available routes and the status of other routing devices, then update the device's routing table with current data.

These routing tables typically exist as proprietary outgrowths of the RTM software. This solution is one of the principal challenges in developing next-generation routers. As device functionality increases, routing table management presents a significant programming bottleneck. Lacking support for the complex data types and multiple access methods that are hallmarks of databases, self-developed routing table management (RTM) structures provide a limited toolset.

In addition, like any data management solution that is hard-wired to the application it supports, routing tables encounter difficulties in extensibility and reliability. Changes made to the data management code reverberate through the entire RTM structure, causing unwanted surprises and adding to QA cycles. Scalability is also an issue: self-developed data management that works well for a given task often stumbles when the intensity of use is ratcheted up. The result is that while the Internet's growth requires rapid advances in routing technology, this device evolution is slowed by software architecture that has outlived its usefulness.

Under such conditions, using a database would seem to be a no-brainer. But deploying a traditional DBMS within an IP router is problematic. Real-time internet address lookups won't accommodate the latency required to go to disk and perform the caching, transaction logging and other processes that are part and parcel of disk-based DBMSes.

In addition, imposing a large database footprint within the router necessitates more RAM and a more powerful CPU. This adds to the overall device cost, and the market for routers is price-competitive. Even a slightly lower per-unit price

increases the manufacturer's market share, and a lower per-unit cost drops right to the bottom line. Software that saves RAM, or requires a less expensive processor, can determine product success.

The emergence of in-memory databases allows the application of DBMS technology to many embedded systems. For developers of embedded systems, proven database technology provides benefits including optimized access methods and data layout, standard and simplified navigation methods, built-in concurrency and data integrity mechanisms, and improved flexibility and fault tolerance. Adoption of this new breed of DBMS simplifies embedded system development while addressing growing software complexity and ensuring high availability and reliability.

Steve Graves is president and cofounder of McObject, developer of the eXtremeDB in-memory database system. As president of Raima Corporation, he helped pioneer the use of DBMS technology in embedded systems, working closely with companies in building database-enabled intelligent devices. A database industry veteran, Graves has held executive-level engineering, consulting and sales/marketing positions at several public and private technology companies.

Archive Index Issue Table of Contents

Advanced search

The Kernel Hacker's Guide to Source Code Control

Greg Kroah-Hartman

Issue #101, September 2002

Greg explains how to use patch and diff or BitKeeper for kernel development.

Many issues involved with Linux kernel development are different from traditional software development processes. When working on a portion of the kernel (or a specific driver), you need to 1) stay aware of changes that are happening to other portions of the kernel with which you interact, 2) constantly apply your changes to the moving target of a fast-based kernel development release schedule, 3) resolve any merge conflicts between changes you have made and changes made by other people and 4) be able to export your changes in a format others can use easily.

For a number of years, I developed and maintained the USB to serial port drivers and then eventually took over maintaining all of the USB code in the kernel. In this article, I explain some of the tools I used in the past to do this work and show how some new tools have enhanced my ability to keep on top of changes in the kernel and let me do my job with less effort.

patch and diff

One of the most common methods of doing kernel work is to use the patch and diff programs. You can use this and no other type of source-code control system to do kernel development. One way is to use two different directory trees: a “clean” one and a “working” one. The clean tree is a released kernel version, while the working one is based on the same released kernel version, but contains your modifications. Then you can use patch and diff to extract your changes and forward port these changes to a new kernel release. For example, let's start off with a clean 2.4.18 kernel (available at www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.18.tar.gz) in our working directory:

```
$ ls
linux-2.4.18.tar.gz
```

Uncompress this kernel, and then rename the created directory, which will be called “linux” to something that makes sense:

```
$ tar -zxf linux-2.4.18.tar.gz
$ mv linux linux-2.4.18
$ ls
linux-2.4.18  linux-2.4.18.tar.gz
```

Now create a duplicate version of this kernel version, and name it something else:

```
$ tar -zxf linux-2.4.18.tar.gz
$ mv linux linux-2.4.18-greg
$ ls
linux-2.4.18  linux-2.4.18-greg  linux-2.4.18.tar.gz
```

Now we can do all of our development in our -greg directory and leave the clean, original kernel directory alone. After we are finished with our work, we need to create a patch to send to other people. The Documentation/ SubmittingPatches file explains the proper format that most kernel developers like for sending and receiving patches. It also explains the usage of a dontdiff file, which can help with generating these patches. The dontdiff file can be found at www.moses.uklinux.net/patches/dontdiff and contains a list of files that you do not want to have the diff program pay attention to.

To create a patch, use the following command:

```
$ diff -Naur -X dontdiff \
linux-2.4.18 linux-2.4.18-greg/ > my_patch
```

This creates a file called my_patch that contains the difference between your work and a clean 2.4.18 kernel tree. This patch then can be sent to other people via e-mail.

New Kernel Versions

If a new kernel version is released, and you want to forward port your changes to the new version, you need to try to apply your generated patch onto a clean kernel version. This can be done in the following steps:

1. Generate your original patch, as in the previous example.
2. Using the official patch from kernel.org, move the old kernel version forward one release:

```
$ cd linux-2.4.18
$ patch -p1 < ../patch-2.4.19
$ cd ..
$ mv linux-2.4.18 linux-2.4.19
```

1. Move your working directory forward one release by removing your patch, then applying the new update:

```
$ cd linux-2.4.18-greg
$ patch -p1 -R < ../my_patch
$ patch -p1 < ../patch-2.4.19
$ cd ..
$ mv linux-2.4.18-greg linux-2.4.19-greg
```

1. Try to apply your patch on top of the new update:

```
$ cd linux-2.4.19-greg
$ patch -p1 < ../my_patch
```

If your patch does not apply cleanly, resolve all of the conflicts that are created (patch will tell you about these, leaving behind .rej and .orig files for you to compare and fix up manually using your favorite editor). This merge process can be the most difficult part if you have made changes to portions of the source tree that have been changed by other people.

If you use this development process, I highly recommend getting the excellent patchutils set of programs (found at cyberelk.net/tim/patchutils). These programs enable you to manipulate text patches easily in all sorts of useful ways, and they have saved kernel developers many hours of tedious work.

Directory Tip

Source Code Control

The process of kernel development using patch and diff generally works quite well. But after a while, most people grow tired of it and look for a different way to work that does not involve so much tedious patching and merging.

A few years ago I discovered BitKeeper (available at www.bitmover.com) and have been using it ever since for kernel development. It originally enabled me to track easily external changes to the kernel tree and allowed me to forward port my kernel changes with almost no effort. Now that Linus Torvalds and Marcelo Tosatti are using BitKeeper for their kernel development, it also allows me to send patches to them easily for inclusion into the main kernel tree.

The use of BitKeeper as a kernel development tool is one that a lot of people find contentious, given BitKeeper's licensing strategy. Read over the license and decide for yourself if you should use it. You also should go through the tutorial on the BitMover web site to familiarize yourself with the tool and some of the different commands.

To do kernel work with BitKeeper, you can base your kernel off Linus' or Marcelo's kernel tree, or you can create your own, with all of the different versions. However, unless you are planning on using BitKeeper to send your patches to Linus or Marcelo, I recommend creating your own kernel tree. That

way you are not buried in the vast number of different changesets that all of the different kernel developers are creating, and you can focus on your work.

Two Trees

Again, with BitKeeper you end up creating two different trees (or repositories as I will now call them) to do kernel work: a clean tree and a working tree.

To create a clean BitKeeper repository, start with a released kernel in your working directory:

```
$ ls
linux-2.4.18.tar.gz
```

Uncompress this kernel:

```
$ tar -zxf linux-2.4.18.tar.gz
$ ls
linux  linux-2.4.18.tar.gz
```

Now create a BitKeeper project called linux-2.4:

```
$ bk setup linux-2.4
```

BitKeeper will ask you a few questions and then provide a file to edit where you should describe your project. Fill this out with your favorite editor, and save it.

You will now have a directory called linux-2.4, which is where your project will be held. Now import the original kernel version into the new repository:

```
$ ls
linux  linux-2.4  linux-2.4.18.tar.gz
$ bk import -tplain linux linux-2.4
```

This will take some time. After BitKeeper is finished importing all of the files, I recommend tagging this point with the kernel version number. This will allow you to find the different kernel versions more easily in the future:

```
$ cd linux-2.4
$ bk tag LINUX_2.4.18
```

Now make a clone of that repository, which is a clean kernel tree, in a different directory so you can make your own changes:

```
$ bk clone linux-2.4 greg-2.4
```

All of our kernel work will be done in the greg-2.4 directory.

You can use the -l option to bk clone. That will use a lot less disk space and go faster by creating hard links to the metadata files. If a file is modified, BitKeeper

will break the link and create a new one where needed. If you end up creating a lot of different repositories on the same disk, you should use this option.

After we are finished with our work, creating changesets by checking in our changes all during the development process (see the BitKeeper tutorial for more details of this), we would like to create a patch to show our changes. This can be done with a simple command from within the greg-2.4 directory:

```
$ bk export -tpatch -rLINUX_2.4.18..+ -h \
> ../my_patch
```

This will create a patch showing all of the changes from the tagged version (LINUX_2.4.18) up to the current changeset and save it in the my_patch file. This patch can then be sent to other people through e-mail, just like any patch created with diff. You will notice that creating this patch was a much shorter process than the previous method of using diff and patch.

Submitting Kernel Patches

New Kernel Versions

When a new kernel version is released, you will want to forward port your changes to the new version. This is where BitKeeper really shines over the previous patch and diff method.

First, go to the original, clean kernel tree and import the new patch:

```
$ ls
greg-2.4 linux-2.4 patch-2.4.19
$ cd linux-2.4
$ bk import -tpatch -SLINUX_2.4.19 ../patch-2.4.19 .
```

If BitKeeper thinks any files that the patch file shows as created and deleted might actually be files that were renamed or moved around the tree, it will pop up a GUI tool that you can use to show manually which files were renamed, which files simply were deleted and which ones simply were created. Figure 1 shows an example of this dialog box.

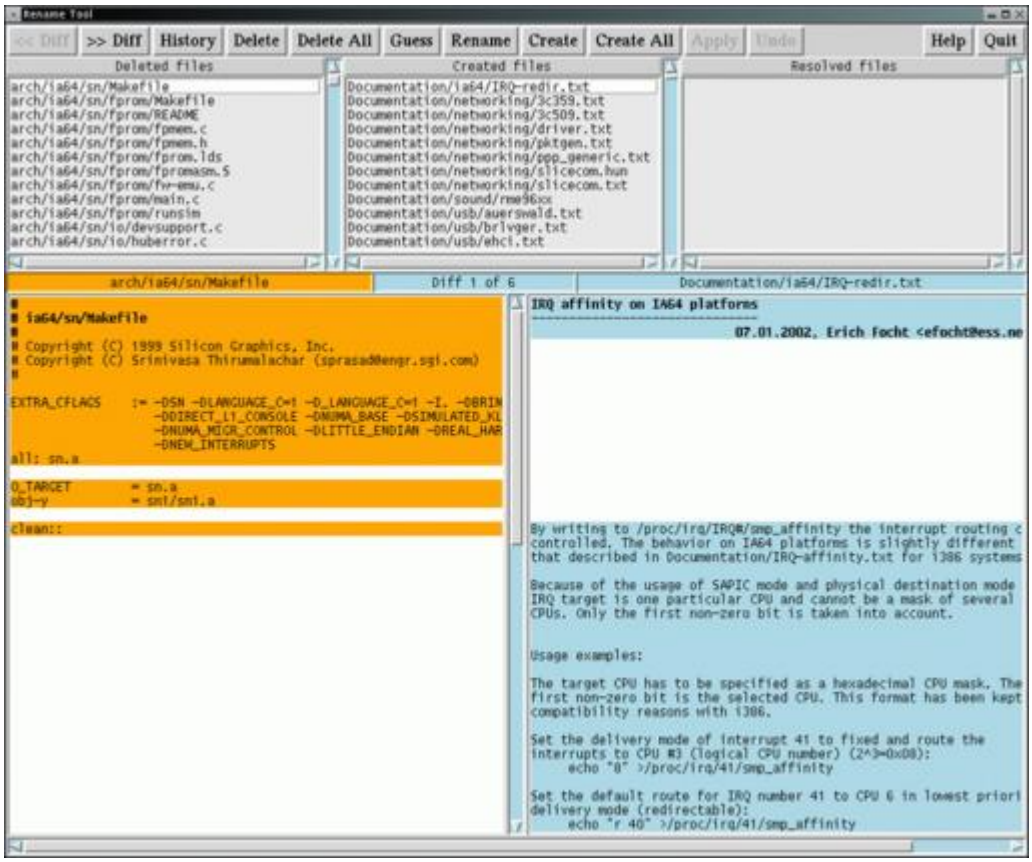


Figure 1. BitKeeper Example Dialog Box

Now go back to your working repository and pull the new changes into it:

```
$ cd ../greg-2.4
$ bk pull
```

BitKeeper will then merge all of the changes between kernels 2.4.18 and 2.4.19 into your working repository. If there are any merge conflicts between any changes you have made and changes that have showed up in the new kernel version, it will report this and ask you what you want to do. I suggest using the graphical three-way merge tool to help resolve these conflicts. This tool shows the original file with the changes that you have made and the changes that the patch (or someone else) has made. It then lets you pick which change you want to accept, or you can hand-edit the file, merging both changes together. Figure 2 shows an example of a change that I made to a file that conflicts with a change that happened in the main kernel.

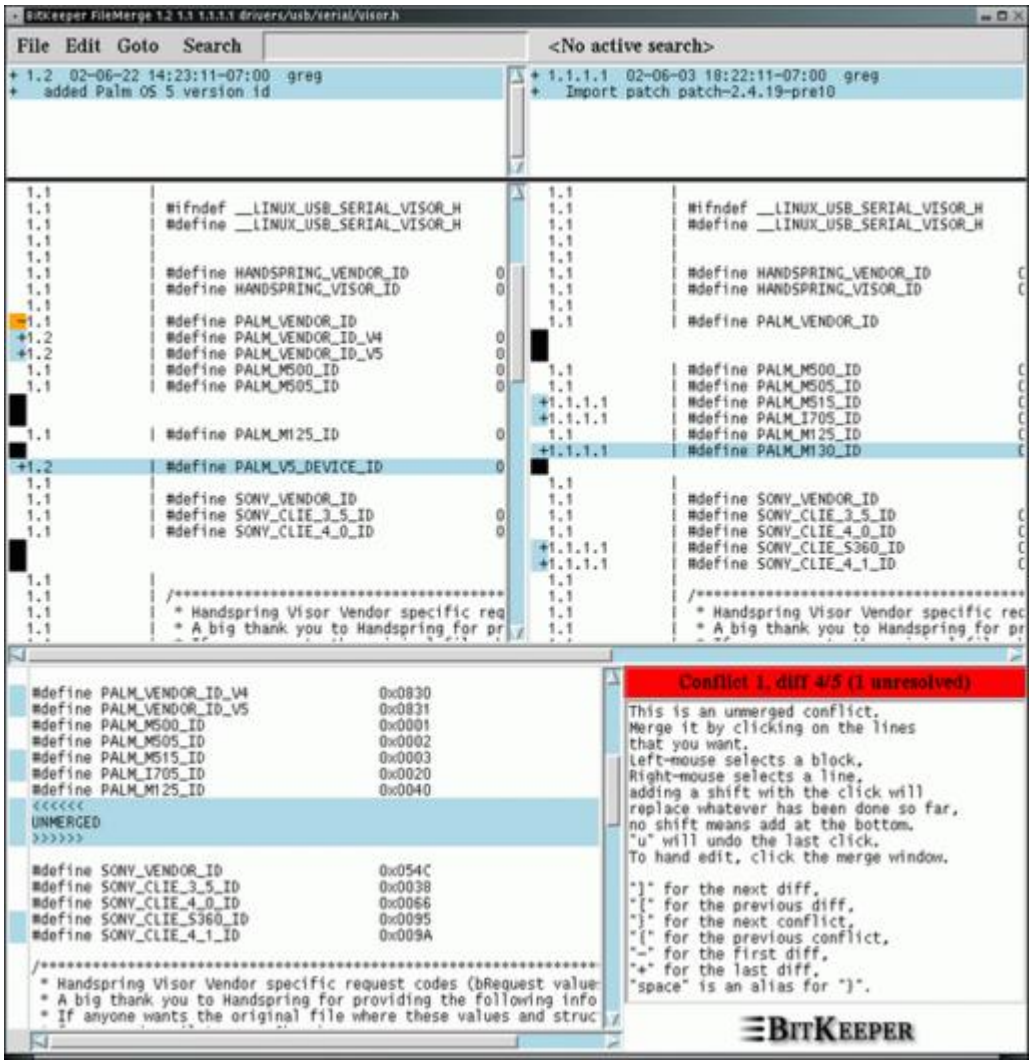


Figure 2. A Merge Conflict

After you are finished resolving any conflicts (and wasn't that much easier than manually looking through .rej files?), you can continue working in your updated kernel. Again, to export a patch with all of the changes you have created, use the following command within the greg-2.4 directory:

```
$ bk export -tpatch -rLINUX_2.4.19..+ -h \
> ../my_patch
```

Other Benefits of BitKeeper

BitKeeper also allows you to see easily all of the changes that have happened to a specific file over time. You can see if the file was modified by one of the main kernel patches or by yourself. An example of the changes that have happened to the drivers/usb/serial/usbserial.c file over time in my repository can be seen in Figure 3. With this tool, you can see what other changes happened at the same time and even what line of code was modified in which version.

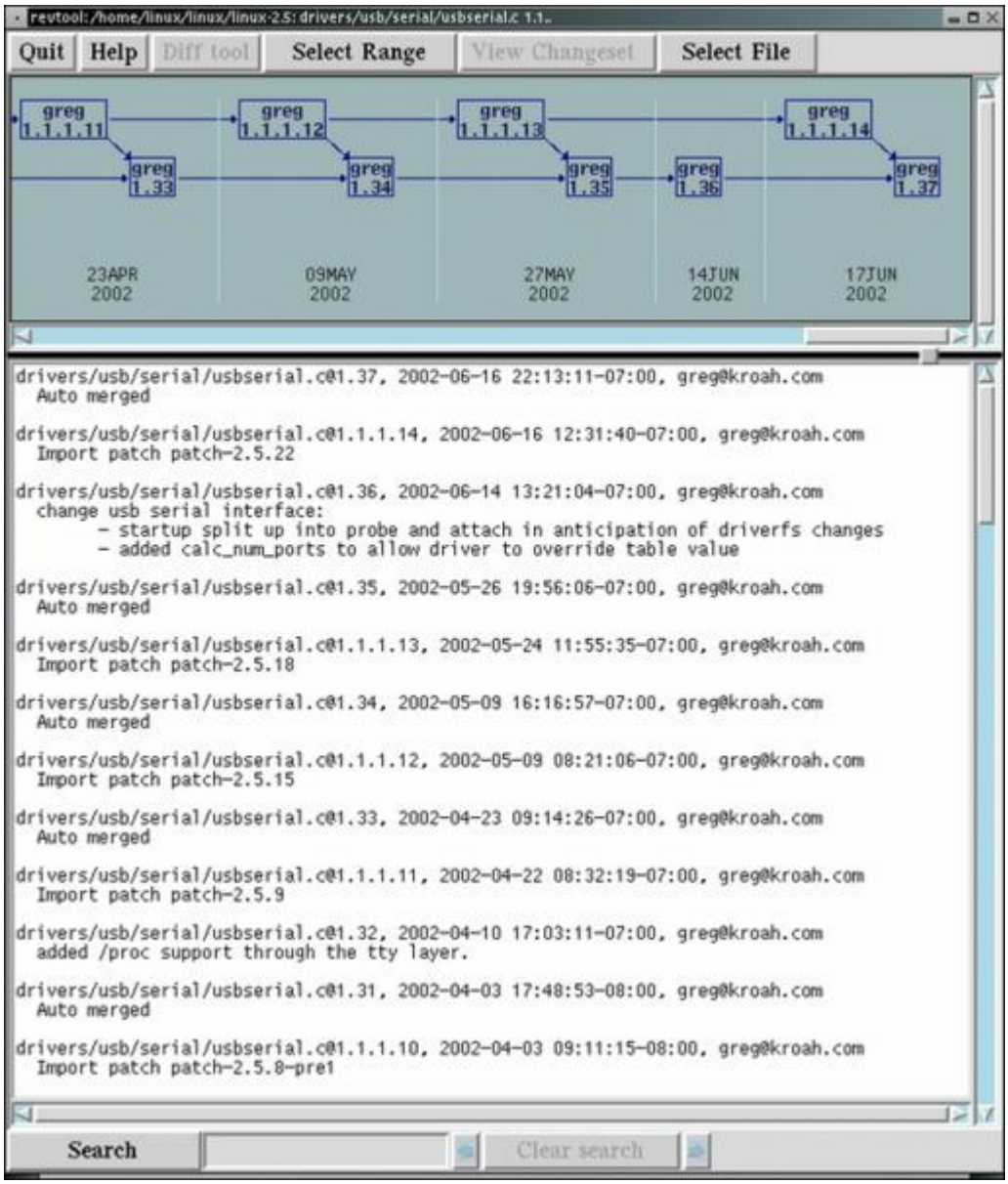


Figure 3. BitKeeper Keeping Track of Changes

One of the strongest benefits of using BitKeeper for your kernel development is that it is a very powerful version control system, and it allows you to work with other developers on the same sections of code at the same time. You can allow other people to pull from your working tree, or you can set up a local server to store your working tree. See the BitKeeper tutorial and documentation for some good examples of how this can be set up and how the development life cycle can be used.

Conclusion

I have shown two different ways of doing Linux kernel development, one with only patch and diff and one using BitKeeper. Personally, BitKeeper has enabled me to spend more time actually doing development work and less time messing with merges. It has also kept me sane in trying to track the 2.2, 2.4 and 2.5 kernel trees for the Linux USB and Linux Hot Plug PCI drivers.

Resources

Greg Kroah-Hartman is currently the Linux USB and PCI Hot Plug kernel maintainer. He works for IBM, doing various Linux kernel-related things and can be reached at greg@kroah.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Introducing AOLserver

Reuven M. Lerner

Issue #101, September 2002

Using AOLserver is not nearly as difficult or challenging as you initially might expect.

Apache, the well-known HTTP server, is something of a poster child for open-source software: it is popular, stable, flexible, secure, portable, extensible and compliant with internet standards. I've been using Apache since it was first released, and it's a joy to work with.

But given that there are multiple open-source operating systems, editors, databases and programming languages, it shouldn't come as a surprise to hear that Apache isn't the only open-source HTTP server. What is surprising is that one of the alternatives comes from America Online, the same company that sponsors Mozilla, the open-source web browser.

AOLserver offers many of the same features as Apache: it is released under an open-source license, is easy and flexible to configure and offers an API for writing plugin modules. But AOLserver has a fundamentally different architecture from Apache, which often makes it a more efficient choice. Moreover, AOLserver includes a built-in Tcl interpreter, multiple threads, database API and database connection pooling. If your web site uses a lot of database connections, then it's worth looking into AOLserver as an alternative to Apache.

This month, we look at AOLserver as a lead-in to several articles about the open-source OpenACS (Open Architecture Community System) web application framework. While AOLserver is not an absolute requirement for OpenACS, it is the standard and expected way to configure and install the system.

History

AOLserver began as NaviServer, written and sold by a pioneering company called NaviSoft that offered high-quality, client- and server-side tools for web publishers. AOL has often bought companies that have developed interesting technology; in the case of NaviSoft, AOL bought them for their server- rather than for their client-side tools.

AOLserver probably would have remained relatively unknown were it not for a combination of events: AOL made binary copies of the program available at no cost via the Internet, and Philip Greenspun began work on the ArsDigita Community System. AOLserver makes it easy to create high-performance, database-backed web sites; the fact that ACS made heavy use of relational databases meant that AOLserver was a perfect fit.

But while AOLserver was free of charge, the source code still was unavailable to the general public. In 1999, Greenspun helped negotiate a deal that resulted in the release of AOLserver 3.0 under the AOLserver Public License, which is essentially identical to the Mozilla Public License.

ArsDigita itself effectively went out of business earlier this year, with its remaining staff and assets becoming part of Red Hat Software. But the original ACS software lives on in the OpenACS Project, which is based on the original Tcl version of ACS, using AOLserver and either PostgreSQL or Oracle.

AOLserver's transition to an open-source model wasn't without its problems. While the details are still a bit sketchy, a number of OpenACS developers briefly worked on their own fork of AOLserver, which they called OpenNSD, arguing that the AOLserver developers needed to be more open to community involvement. OpenNSD now appears to be dead, with the OpenACS community once again encouraging people to use AOLserver.

At the same time, ArsDigita added a number of enhancements to AOLserver that have not yet been added to the server's source code in its latest stable or development editions. Because my use of AOLserver is almost always connected to OpenACS, I will be using AOLserver 3.3ad13, available at www.openacs.org/software.adp. More advanced versions, including a development snapshot for the upcoming 4.0 version, are available at www.aolserver.com. But right now, these official versions are not guaranteed to support OpenACS.

What Makes It So Great?

Until earlier this year, when version 2.0 was first released to the general public, Apache was a multiprocess server. That is, a number of Apache processes

would run at any given time, with each process able to handle a single HTTP transaction at a given time. Supporting ten simultaneous transactions meant having ten Apache processes running in parallel, while supporting 100 simultaneous connections meant having 100 such connections available.

Apache 2.0 changes this somewhat, allowing you to have multiple threads per process. Each thread can handle an HTTP connection, meaning that five threads in each of five processes can accept up to 25 simultaneous connections. Because threads typically consume fewer resources than processes, this effectively means that a typical PC will benefit from an increase in performance.

AOLserver, by contrast, has always been multithreaded and operates within a single process. At any given time, your computer will be running only one copy of nsd, whose name reflects the fact that it was once the NaviServer dæmon. But that one process can handle a large number of simultaneous HTTP connections. Indeed, AOL continues to push AOLserver development precisely because it can handle such a large number of simultaneous connections. It uses AOLserver on its own high-traffic web sites, including netscape.com, aol.com and digitalcity.com.

One advantage—and potential pitfall—of a multithreaded solution is the fact that the threads easily can share data structures with one another. AOLserver takes advantage of this to create a pool of database connections. Because connections in this pool are always kept open, your web applications will not have to spend time opening (or closing) them. Moreover, because it is rare for all of a server's current HTTP connections to require simultaneous access to a database, the pool can contain fewer connections than the maximum number of threads—reducing the amount of memory used by the web server and the database server. You can think of this as the database equivalent of packet switching, in which a telephone line is shared among numerous parties by taking advantage of the fact that no one needs the line 100% of the time.

AOLserver supports plugin modules, much as Apache does. There are a number of modules available, ranging from an XML parser (nsxml) to an embedded version of Python (PyWx). There are also modules for executing CGI programs, for secure connections via SSL and for individual relational databases such as MySQL, PostgreSQL and Oracle. Because OpenACS can work with either PostgreSQL or Oracle, the version of AOLserver available for download from openacs.org includes both of these modules, as well as nsxml.

Just as mod_perl allows web developers to customize Apache's configuration and responses without using C, AOLserver provides an integrated API that allows you to create custom functionality in the Tcl language. Truth be told, I

personally would prefer to use Perl or Python for development, but as many AOLserver and OpenACS developers have told me over the years, Tcl “isn't that bad”, and I have managed to create a number of interesting, maintainable applications using Tcl and AOLserver in a remarkably short period of time. (I haven't yet tried the embedded Python module, in part because OpenACS requires the use of Tcl.) And, the API that AOLserver provides makes it relatively easy to work with such things as HTTP headers and HTML form values.

Compiling and Configuring

Compiling AOLserver is relatively straightforward. Unlike Apache, which provides support via the apxs program for compiling modules after you have installed the server, AOLserver requires that all modules be compiled and installed together.

While creating this user and group is not mandatory, AOLserver will refuse to run as root for security reasons. So before you begin to compile and install AOLserver, you should create a new user and group on your system, traditionally called nsadmin. On my Red Hat 7.2 system, I simply say:

```
/usr/sbin/adduser nsadmin
```

While still logged in as root, I now create the /usr/local/aolserver directory, into which AOLserver is installed by default. I then give ownership of this directory to nsadmin:

```
mkdir /usr/local/aolserver
chown nsadmin.nsadmin /usr/local/aolserver
```

Once we've done this, I change to the nsadmin user, open the source code that I downloaded from openacs.org and begin the compilation process:

```
su - nsadmin
cd /tmp
tar -zxvf aolserver3.3ad13-oacs1-beta-src.tar.gz
cd aolserver
./conf
```

This will automatically configure, compile and install AOLserver according to your system's parameters, placing files under the directory /usr/local/aolserver. AOLserver automatically will try to compile whichever modules it can, ignoring (and excluding) any others. On my desktop machine, which has development libraries for PostgreSQL but not for Oracle, configuring and installing AOLserver in this way results in the inclusion of the PostgreSQL driver, but ignores the Oracle driver entirely.

The build process can take awhile and doesn't produce a great deal of output to the screen. If you are concerned that the process has somehow become

frozen, you can look at the log/aolserver.log file; all of the compilation output can be found there.

When the build process is done, you will have a copy of AOLserver in /usr/local/aolserver. The most important directories under /usr/local/aolserver are bin, in which the AOLserver (nsd) program is located, along with the shared libraries (.so) for each of the modules that were compiled into the server. The log directory contains access and error logs for the server, and the lib directory contains the built-in Tcl interpreter.

AOLserver's configuration file is written in Tcl; a simple configuration file is placed by default in /usr/local/aolserver/sample-config.tcl. If you examine it, you will see that each of the configuration directives is actually a Tcl variable assignment. These variable assignments are divided into sections, where each section normally represents a module that was compiled into the server.

As you can see from the sample configuration file, you can assign literal values to variables. For example, you can set the HTTP port to which AOLserver listens to 8000 with the following:

```
set httpport 8000
```

Because the configuration file is written in Tcl, you also can set the homedir variable, which is /usr/local/aolserver by default, by asking AOLserver rather than hard-coding the value:

```
set homedir          [file dirname [ns_info config]]
```

And of course, you can base variable settings on the values of other variables, using simple interpolation:

```
set servername             "server1"
set pageroot ${homedir}/servers/${servername}/pages
```

Those two lines, from the sample configuration file that comes with AOLserver, configure the root of static URLs to be in /usr/local/aolserver/servers/server1/pages.

Other configuration settings are made with the ns_param command, which typically takes two parameters: a name and a value. Each parameter must come in a section, begun by a call to ns_section. For example, we can activate server debugging by turning on the debug parameter in the (global) ns/parameters section:

```
ns_section "ns/parameters"
ns_param   debug           false
```

Unfortunately, the documentation for AOLserver's parameters is quite lacking when compared with the on-line Apache documentation. An almost complete list of parameters is at aolserver.com/docs/admin/config-reference.tcl.txt, which demonstrates a server configuration that sets nearly everything.

Once you have finished configuring your system—and the default configuration is a good start for simple sites—you can start AOLserver by invoking the nsd program and specifying the name of the configuration file you want to use:

```
cd /usr/local/aolserver
bin/nsd -f -t sample-config.tcl
```

The -f option runs AOLserver in the foreground, sending the error log to your screen. Once you feel comfortable with what's happening, you can remove the -f, looking in the log directory for your server's error log.

If you want AOLserver to listen to port 80, then you must start it as root. Otherwise, Linux will refuse to honor the request, telling you that only the superuser can start servers that listen to “privileged” ports (i.e., less than 1024). If only root can listen to port 80, but AOLserver refuses to run as root, how can you serve port 80? By starting AOLserver as root and passing it options to indicate the user and group to which it should switch:

```
su root
cd /usr/local/aolserver
bin/nsd -f -u nsadmin -g nsadmin -t \
  sample-config.tcl
```

You should now be able to point your browser at http://yourhost.yourdomain.com:8000/ and see the introductory AOLserver document, welcoming you to this new server. Note that AOLserver's configuration file looks for both your computer's name and its IP address, so if you are connected to a network, you will not be able to point your browser to localhost, but will instead need to use its full name.

Tcl Programs

While AOLserver is undoubtedly an excellent HTTP server for static documents, you're unlikely to use it for that. It's far more common to create dynamic pages using Tcl.

The easiest and simplest way is to create a Tcl program that returns HTML. To do this, create a file in the pageroot that ends in the .tcl extension. It can create any Tcl that you want; the most important thing, however, is that it end with ns_return—a Tcl procedure defined by AOLserver that takes three arguments: 1) a numeric HTTP response code (such as 200 or 404) that indicates the success or failure of the program's execution, 2) a “Content-type” header that

describes the type of content that is being returned and 3) the actual content to return to the user.

For example, here is a simple “Hello, world” program:

```
ns_return 200 text/html "<html>
<head>
    <title>Testing</title>
</head>
<body>
     <p>Hello, world</p>
</body>
</html>"
```

If you stick the above program in the pageroot directory as hello.tcl and load it into your browser, you will get the literal contents of the file returned to you. That's because we need to reconfigure AOLserver to allow .tcl pages within the pageroot. We do this by setting the enabletclpages parameter to true within the ns/server/${servername} section:

```
ns_section "ns/server/${servername}"
ns_param   enabletclpages  true
```

Once you have made this change, you can restart AOLserver and retrieve hello.tcl once again. This time, you should see HTML output rather than the verbatim, text/plain output.

A .tcl page can do a host of different things: connecting to a database, retrieving information from XML files to retrieving information from across networks or receiving information from an HTML form. Because Tcl comes with a variety of string-manipulation commands, you can parse the input and interpolate variables into the output in a wide variety of ways.

Note that the Tcl is interpreted within AOLserver, rather than as an external process. This means that such .tcl files execute much faster than a CGI program would; in many ways, they run similarly to Perl programs in mod_perl.

It's true that .tcl files are great when a programmer is creating the outgoing HTML. But as I (and others) have learned from bitter experience, graphic designers generally are unprepared to modify HTML that appears within source code files. For this reason, many web developers have switched over to templates—be they ASP, JSP, HTML::Mason, DTML or a variety of other similar technologies. AOLserver comes with its own built-in templating system, known as ADP (AOLserver Dynamic Pages), whose syntax is suspiciously similar to Microsoft's ASP. Code that you want to execute goes within <% and %>, while the code that outputs a value into the surrounding HTML goes within <%= and %>. For example:

```
<% set foobar "abcdef" %>
<head>
    <title>Testing</title>
</head>
<body>
    <p>Hello, world</p>
    <p>Hello, <%= $foobar %></p>
</body>
</html>
```

Sites larger than a few pages probably will want to share some Tcl code. The easiest way to do this is to create one or more .tcl files that define procedures that AOLserver will load at startup time. These procedures then will be available to all of your .tcl and .adp pages. To enable this functionality, we must add the following to our sample-config.tcl file:

```
ns_section ns/server/${servername}/tcl
ns_param   library \
  ${homedir}/servers/${servername}/tcl
ns_param   autoclose      on
ns_param   debug          true
```

Since our server name is server1, any .tcl file that we place in /usr/local/aolserver/servers/server1/tcl will be loaded when AOLserver starts up. For example, I added the file foo.tcl to that directory, whose contents consisted of:

```
proc return_hello {} {
    return "hello"
}
```

I restarted AOLserver (which is necessary in order for it to read Tcl library files) and modified hello.adp to read:

```
<% set foobar "abcdef" %>
<% set hello [return_hello] %>
<head>
    <title>Testing</title>
</head>
<body>
    <p>Hello, world</p>
    <p>Hello, <%= $foobar %></p>
    <p>Hello, <%= $hello %></p>
</body>
</html>
```

Now the value of the “hello” variable is set to the output from my return_hello proc, which in this case is nothing more than the word hello.

Because you must restart AOLserver in order for it to load library Tcl procedures, I've often found it easiest to define new procedures within <% %> sections in my ADP pages. Once I see that the procedure works correctly, I move its definition to the library directory and restart AOLserver only once.

ADP and .tcl pages are fine for documents that contain some dynamic content. But sometimes you want to associate programs with a URL without necessarily creating a document on disk. We easily can take care of this by defining a Tcl procedure in our library file and then registering that procedure with a particular URL using the AOLserver ns_register_proc command:

```
proc http_hello {} {
  ns_return 200 text/html "<html>
                           <head><title>Hello!
                             </title></head>
                           <body><p>Hello!
                             </p></body>
                           </html>"
}
ns_register_proc GET /hello http_hello
```

If you put this Tcl code in a file that sits within the library directory we defined earlier, restart your server and point your browser to /hello, you will see the output from our http_hello procedure.

I've been programming in mod_perl for several years and am still impressed by the ease with which you can create new dynamic pages with AOLserver and ns_register_proc. Moreover, you can register different procedures for GET and POST requests. You even can register filter procedures that can monitor or change the output generated by another page.

Conclusion

If you plan to use OpenACS for on-line communities, then you almost certainly will have to learn how to work with AOLserver. But even if OpenACS does not interest you, AOLserver's flexibility, speed and multithread capabilities are well worth investigating for your dynamic web sites.

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 2002 by Prentice Hall. Reuven lives in Modi'in, Israel, with his wife and daughter.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The Ultimate (but Small) Linux Box!

Marcel Gagné

Issue #101, September 2002

You don't always need the biggest, baddest machine to get the job done well—slim down the numbers.

Great meals are made in great kitchens, François, but what makes a kitchen great is simple. A great kitchen is a marriage of talent and environment, the chef and his or her tools. To do wonderful things, one doesn't need the world's largest kitchen any more than one needs the biggest and fastest computer when cooking with Linux. Observe, *mon ami*, this window manager, IceWM. While it is a small package, it is still quite attractive, flexible and easy to work with.

François, you are not paying attention. *Quoi?* Ah, welcome, *mes amis*, to *Chez Marcel*, home of fine Linux cooking. Please, let me show you to your seats while François fetches the wine. *Vite*, François. Slip down into the north wing of the wine cellar and bring back that 1999 Napa Valley Cabernet Sauvignon we were, uhm, submitting to quality control earlier today.

While we wait for the wine to arrive, let me tantalize you with some hints from today's menu. It is easy to get carried away with the idea of a super fast processor (or several), vast amounts of disk space and virtually endless memory. Unfortunately, that is not the machine we are all blessed with. In fact, those who might remember the days of being a poor student certainly will appreciate that, sometimes, we take what we can get. How, then, does someone with only modest hardware take advantage of the power of Linux?

For starters, we could run a very small system by going to some Linux distributions on a single floppy and ignoring the graphical environment entirely, but I would like to avoid doing that. The idea is to create an attractive desktop with some friendly graphical tools while living within the constraints of limited resources.

Here are some interesting numbers: on my new Red Hat 7.3 workstation, running the KDE 3.0 desktop with a single xterm, the **free** command shows roughly 34,000KB (not counting buffers and cache) of memory in use. This is after I subtract the base system requirements and without running a login manager, such as KDM. Granted, a lot of KDE's funky new features are running, such as fading tool tips, pop-up icons, sound themes and so on, but that is the default install. GNOME 1.4, again with a single xterm, comes in closer to 27,000KB. Those KDE and GNOME numbers present a sharp contrast to what I get when I run this IceWM—a mere 7,500KB.

As you can see, you can make a huge difference right from the start even when continuing to run a graphical environment. IceWM, written by Marco Macek and enhanced by Mathias Hasselmann, is a small, lightweight window manager that is nonetheless feature-rich. It supports multiple desktops or workspaces, can be used without a mouse, supports (and comes with several) themes and does a nice job of mimicking the style of that other OS.

The latest incarnation of IceWM can be found at www.icewm.org. The site provides precompiled RPMs, but building this little window manager is easy and follows the classic extract and build five-step method:

```
tar -xzvf icewm-1.0.9-2.tar.gz
cd icewm-1.0.9-2
./configure
make
su -c make install
```

You no doubt will want to customize IceWM to your own environment and add those little touches that make you feel at home. The best way to do that is to create a local .icewm directory (in your home directory), then copy the system-wide configuration files there. The default installation puts them in /usr/X11R6/lib/X11/icewm.

```
mkdir $HOME/.icewm
cp -r /usr/X11R6/lib/X11/icewm/* $HOME/.icewm
```

Et voilà! You are ready to run your new window manager. The easiest way to get started is to create an .xinitrc file in your home directory. All you really need in that file is a single line that reads:

```
exec icewm
```

Now, type **startx**, and you are skating on the Ice window manager (a little joke, *mes amis*). Click the application launcher button in the lower left-hand corner, start a few programs, and you should have something that looks like Figure 1.

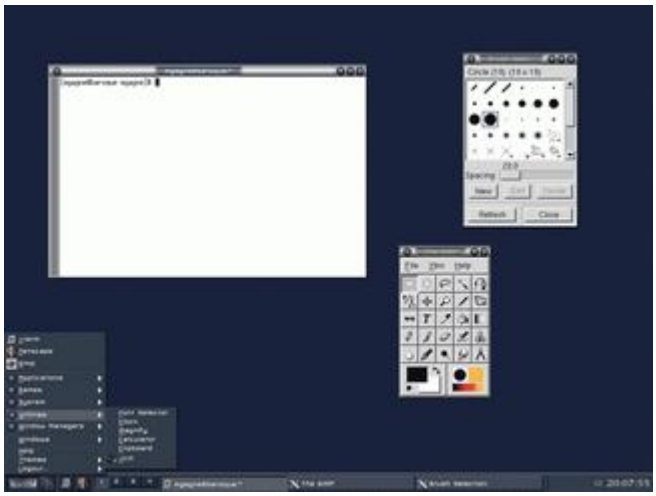


Figure 1. IceWM, a Very Cool Window Manager

After doing some rather casual mathematics, my running instance of IceWM (with nothing but an xterm) comes in at around 7,500KB. Not bad, but where else can we do a little trimming? How about the graphical web browser? It seems lately that web browsers keep getting bigger and bigger. Granted this is because of their increasing richness of features, but on less than super fast hardware, we might be willing to give up a feature or two. Even Opera, an excellent browser that has gotten a lot of press lately (in part thanks to its speed and small footprint), still might be a bit hefty.

Consider Jorge Arellano Cid's Dillo browser as an alternative to the larger, flashier, do-it-all browsers of the day. You'll not only find that its demands on your system are few, but its speedy rendering of pages is also nice. Have a look at Figure 2 for a rather snappy shot of Dillo in action.

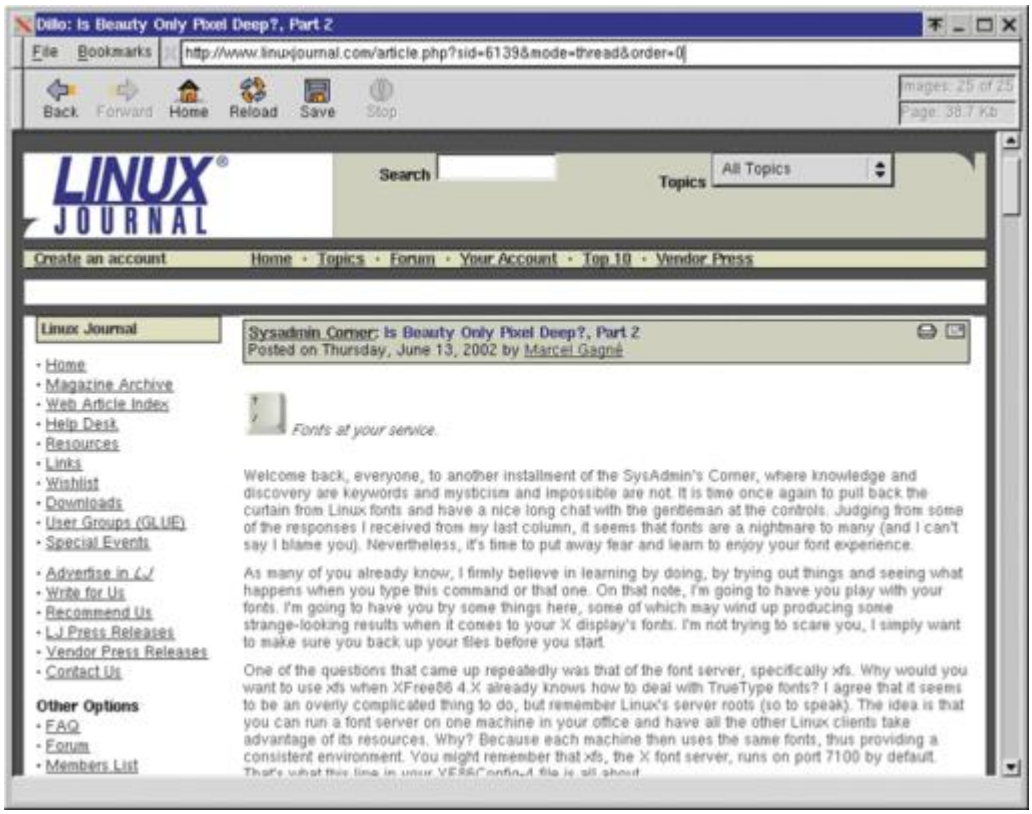


Figure 2. Dillo, a Small and Fast Graphical Web Browser

To get started with Dillo, visit dillo.cipsga.org.br and pick up the latest source. Extracting and building it involves a familiar set of steps:

```
tar -xzvf dillo-0.6.6.tar.gz
cd dillo-0.6.6
./configure
make
su -c make install
```

Once the program is built and installed, you start it by typing **dillo &**. After starting Dillo, I checked my resources yet again and found that the whole running process took only 848KB of RAM, a rather impressive little number.

Arguably, the word processor is the single most important desktop application in the office. What could we possibly do in terms of a full-featured, powerful word processor that handles the ubiquitous MS Word format documents? Keeping something like this small is a tall order, I agree, but how does the 1,700KB of AbiWord sound? Compare that with the nearly 12,000KB my system reports upon firing up OpenOffice Writer. If all you need is a word processor, and your resources are limited, visit www.abisource.com, where you can get yourself a free copy of this great word processor.

A number of prebuilt binaries are available on the site, so you probably don't have to do much work. For those who prefer building from source, you'll have a couple of additional steps, but it is all very simple:

```
tar -xzvf abiword-1.0.2.tar.gz
cd abiword-1.0.2/abi
./autogen.sh
./configure
make
su -c make install
```

When you have finished installing AbiWord, you can start it by typing the command **abiword &**. A word of warning: a common complaint when starting up AbiWord has to do with fonts. By default, AbiWord's fonts aren't included in your X font server's list of available fonts. Consequently, AbiWord will complain on starting up. The program should still run, but you won't have access to the included fonts. To rectify this complaint, you can simply add your fonts to your X font server's font path. On a Red Hat or similar system, the easiest approach is to use the chkfontpath command:

```
chkfontpath -a /usr/local/share/AbiSuite/fonts
service xfs restart
```

The path above assumes that you installed AbiWord from source. On other systems, you may have to edit the /etc/X11/fs/config file and manually add the font path. Look for the paragraph that begins with catalogue =:

```
catalogue = \
/usr/X11R6/lib/X11/fonts/75dpi:unscaled,
```

Notice that each line has a comma at the end of it except for the last. If you are manually adding the font path to the end of this list, make sure you add a comma to the now second-to-last line and remove the comma from the last. Once again, restart the xfs service and then restart your window manager session.

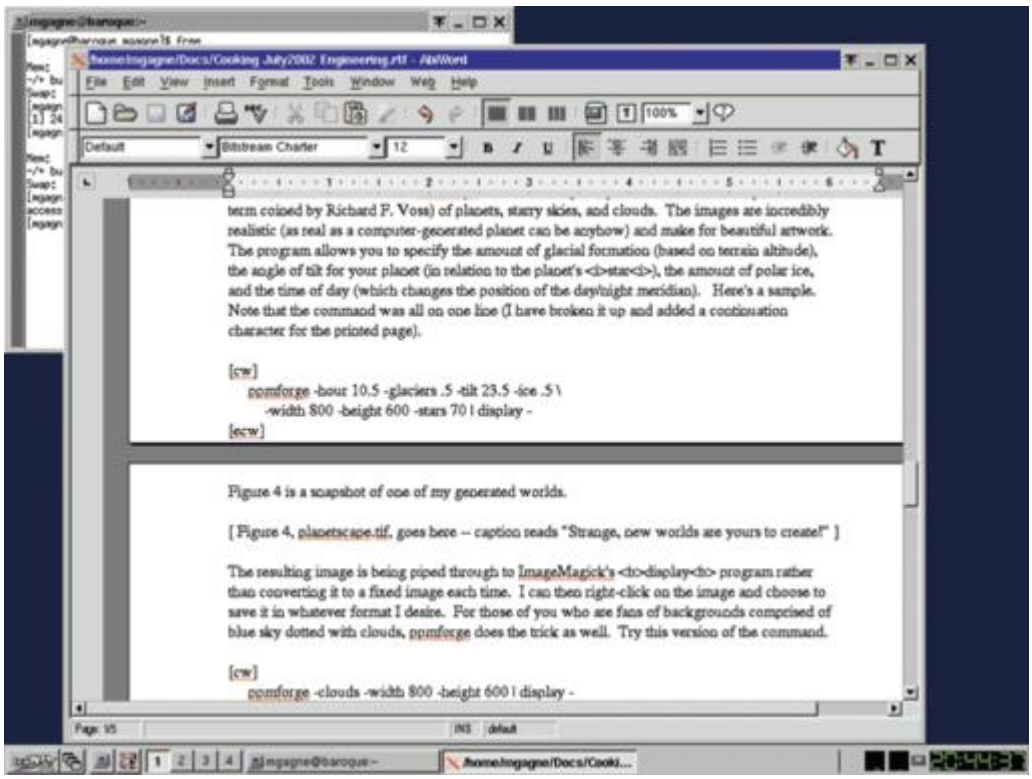


Figure 3. Working with AbiWord

So, we have a tiny but slick window manager, a tiny but slick web browser and a modest but powerful word processor. What about an entire office suite? Once again, I can see from the look in your eyes that you think Chef Marcel has been sampling a little too much of his own wine.

Have a look at Siag Office, whose name stands for “Scheme in a grid”. Not a very intuitive name for an office suite, I grant you, but Siag is an office suite you owe to yourself to have a look at. It is a tightly integrated suite comprised of a word processor (which the author calls Pathetic Writer), a spreadsheet (Scheme in a grid), an animation program (Egon) and more. Right up front, I will tell you that MS Word format is not directly supported. For some, this may be a showstopper, but for others it is less of a problem. Because RTF format can be used to exchange documents, this may be all you need to move documents back and forth.

You'll need to get and compile the XawM libraries (an Athena-compatible library), the Mowitz libraries (more widgets) and finally, Siag Office itself. Trust me, it's easier than it sounds; all of these packages are available on the main site. Each package can be compiled with the classic extract-build five-step method. For instance, with the XawM libraries, you would do this:

```
tar -xzvf XawM-1.5u.tar.gz
cd XawM-1.5u
./configure
make
su -c make install
```

Once you have made and installed all three packages, you should be able to start the word processor by typing the command **pw** and the spreadsheet with **siag**.

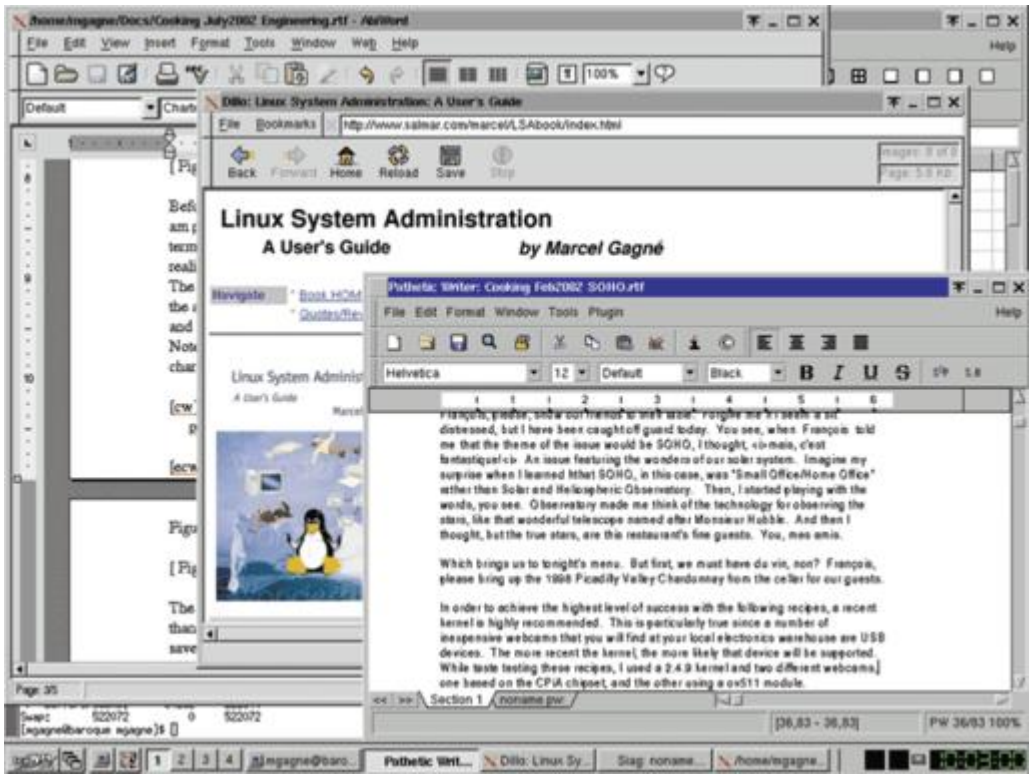


Figure 4. All This for So Little

We've done this little number comparison a few times now, so let's have a look at how much Pathetic Writer demands of my system. Drumroll please. With Pathetic Writer going and a column-length article loaded, I am using a mere 1,300KB.

In fact, with IceWM, AbiWord, Pathetic Writer, the Siag spreadsheet, the Dillo web browser running and an xterm, my memory usage is still under 32MB! Have a look at Figure 4 for a screenshot of my lightweight but busy session.

But, of course, when working with limited resources, applications are only part of the picture. A default installation of any major distribution likely will have a number of unneeded services running. You must ask yourself whether you need to be running things like sendmail, NFS, the Apache server and so on when your PC is being used as a workstation. Run a **ps axfw** and decide whether you need all those services.

As you can see, you don't need to have the biggest, fastest and most up-to-date computer to experience the *ultimate* Linux system. What you need is the willingness to look beyond the most popular packages of the day and ask yourself whether you really need all those features. Sometimes smaller is better. When it comes to wine, however, a nice generous glass doesn't hurt. That large glass can, however, affect your performance (consequently taxis will be waiting outside the restaurant). Drink up. Enjoy. Until next month. *A votre santé*! *Bon appétit*!

Resources

Marcel Gagné lives in Mississauga, Ontario. He is the author of Linux System Administration: A User's Guide (ISBN 0-201-71934-7), published by Addison-Wesley (and is currently at work on his next book). He can be reached via e-mail at mggagne@salmar.com.

Archive Index Issue Table of Contents

Advanced search

Q&A with Chris Wysopal (Weld Pond)

Mick Bauer

Issue #101, September 2002

Chris discusses the problem of application security and the importance of designing products securely from the start.

One of the most interesting, accomplished and productive hacking organizations of the mid- to late-1990s was L0pht Heavy Industries, a loose affiliation of “gray-hat” (i.e., mostly benevolent) hackers. During those years, the L0pht earned worldwide notoriety plus the ire of Microsoft for discovering and publicizing a number of software vulnerabilities, especially in Windows. Combined with the success of their password-auditing tool, L0phtCrack (which, besides exposing poorly chosen passwords also demonstrated inherent weaknesses in early Windows NT authentication implementations), the L0pht's relentless exposure of poor security programming played a significant role in Microsoft's slow but pronounced improvement in addressing security flaws in their products.

The L0pht's fame and popularity culminated in eight of their core members being invited to offer expert testimony on internet security to the US Senate in 1998. One of those members was Chris Wysopal, aka Weld Pond, a veteran

computer security engineer, researcher and programmer. Chris, along with many of his former L0pht colleagues, now works for the consulting firm @stake, with whom L0pht Heavy Industries merged in January 2000.

Chris graciously interrupted his busy schedule as @stake's director of research and development to submit to a Paranoid Penguin interrogation. True to the L0pht's old form, his answers were frank, extremely well informed and thoughtful.

Mick Many of our readers are familiar with your work with the L0pht, but you've been in the public eye a bit less lately. Could you describe your current job at @stake and how it's different from what you were doing before?

Chris I am the director of research and development at @stake. From a management standpoint, I oversee the different research and tools projects that the consultants and developers are undertaking. Areas of research are forensics, attack simulation, wireless and applications. Personally, I have been most involved in the area of application security.

There are actually similarities with the L0pht. Each person has their own area of expertise and is given the opportunity to work on technology that interests them, whether it be tools development or vulnerability research. The difference with @stake is all the research and tools we build have a business need. Most are born out of problems we see working with our customers or grow out of the need to automate security tests we do manually as part of our consulting practice.

Mick What are some of the technologies you've worked with lately?

Chris Recently, I've been working on the problem of application security. How do you design products securely from the start? How do you implement them using secure coding techniques? How do you test that they are secure? This is a difficult problem to solve because you have to fit it into the way software is built in the real world: with a limited budget and extreme time pressure.

The solution we've come up with is to build security into the different stages of the development process. We have come up with techniques for diagramming the threat paths of an application to use during the design process. This allows us to find design flaws efficiently and, at the same time, make sure we have the whole design covered to a certain depth. The next step is building tools to do this.

For the implementation process, we've built tools that model an application's behavior by analyzing the source code or even the binary. This is semantic

analysis and not the simple lexical analysis we did previously with SLINT [a code-auditing tool developed by the L0pht]. It allows automated detection of bad code that will cause buffer overflows or script injection, for example. Dildog, also from the L0pht, and Tim Newsham deserve the credit for this tour de force tool.

To enable automated security testing we're working on application penetration test tools that can fuzz (send random problematic data) arbitrary application protocols such as HTTP. We can set up an application in a lab environment and launch automated attacks. These tools are great for finding buffer overflows, format string, canonicalization and script injection problems. Other tools are shims and proxies to manipulate an application as it reads and writes data over the wire, a system call or an RPC call. I hope these types of tools become part of the standard quality-assurance process that people follow before releasing software.

Mick Indeed. Speaking of software, do you still find much time for coding? Anything in the works you care to discuss?

Chris I haven't found time to do any substantial coding lately. Mostly I'm creating proof-of-concept code or scripts to try out a particular application attack. If I had to mention one cool thing to look out for, it is definitely our source and binary semantic security analysis tool. This is going to bring a revolution in the ability to detect security problems before (and after) a piece of software is released.

Mick Do you see any improvement in the software industry at large in making security a design goal of programming projects rather than an afterthought?

Chris Yes, definitely. The secure software development techniques and tools we have been working on have been well received by our software customers. A lot of this is due to being beaten up about insecure products over the last few (many?) years. Sophisticated technology customers simply are not accepting it anymore. It is becoming part of the purchasing decision.

Another reason things are changing is companies are learning that it's very expensive to patch vulnerabilities after the fact, not to mention the PR nightmare. They are finally realizing that there are people out there actually downloading the trial versions, breaking them in their labs and publishing what they find. [Software companies] just can't hide their shoddy security anymore. Plus, it is cheaper to build in security upfront. We've built a “return on security investment” model using the vulnerabilities and the cost to fix them from the data of 45 customer engagements. The numbers crunch down to a 21% savings

by starting out building a secure application rather than trying to bolt security on after shipping.

Mick The Open Source world has had its share of security crises in the past few months, with a string of vulnerabilities in Secure Shell, Squid, SNMP and zlib, to name a few. Yet some of these affected packages, particularly OpenSSH, are maintained by the OSS community's best and brightest. Is this trend simply bad luck? Is software becoming unsecurably complex, or do you see some other explanation?

Chris Yes, some of it is very complex. SSH took a big leap in complexity going up to version 2.0. I think the code auditing that has gone on has eliminated much of the low-hanging fruit vulnerabilities from important applications. The closed-source vendors are playing catch-up here. But I don't think this eliminates nearly all the problems.

There needs to be an effort to do more than code auditing. There is a need for threat modeling of the designs and application penetration testing such as fuzzing. Some problems, like the ASN.1 problems that plagued SNMP, are difficult to find through auditing. The data paths are becoming very complex. I also think vulnerability researchers are getting better, and there are more people doing it.

Mick Your @stake compadre, Dr. Mudge, has been speaking lately about risk management and other less-technical approaches to IS security. What sort of progress do you see companies and organizations making toward demystifying IS security in this way and institutionalizing good security policies and practices?

Chris The business managers in a company need to understand the risks of not having adequate security. This understanding should not reside in IS alone. Once security-incident costs are quantified for the people in charge of profit and loss, they start to see the value of security and are willing to pay for it. Then people running the business will have a much larger budget to allocate toward security products and services. Once executives in a company are educated to the risks, there is a much better chance that security practices and policies will be adopted company-wide.

Mick What are some approaches that seem to work in getting organizations to adopt better security policies and practices, especially in selling these concepts to nontechnical managers?

Chris Demonstrations work wonders. It's one thing to tell a nontechnical manager that his upstream internet connection can be sniffed and that his company is sending sensitive information in the clear. It's another to show him

the finance department's latest salary updates that they just sent via e-mail to an out-sourced payroll company. It hits home when you get the data. Nontechnical people have a problem understanding what could be done with a vulnerability. It's too hypothetical.

Mick That's certainly my experience too. By the way, it occurs to me that your @stake colleague Frank Heidt will be performing exactly that kind of demonstration on the Linux Lunacy Cruise this October (not that I'm *shilling* this fine *Linux Journal*-sponsored event or anything). Here's a question that is of the utmost importance to our readers: what is your own experience with Linux?

Chris Heh, I am an old-timer. I set up the first Linux box we had at the L0pht in 1994. I think it was the 0.99.pl14 kernel running on a 486. I configured it to be our internet gateway, routing our class C over a 28.8. I was running NCSA web server and sendmail. For a trip down memory lane, check out the L0pht web site as it was running on that box: web.archive.org/web/19961109005607/http://l0pht.com.

We used DESLogin because these were the pre-SSH days. Linux was my first experience with UNIX programming. I had access to a SunOS 2.4 system, but it didn't have the development tools that Linux did. Linux excelled as a learning environment then, as it does now.

Mick What do you see as being some of Linux's strengths and weaknesses from a security standpoint?

Chris Linux has a simpler security model and configuration than many other OSes, although things have been growing in complexity over time. If you aren't doing anything too complex, this simplicity is a big plus. Most of the complexity of other systems ends up shooting programmers and administrators in the foot. Fewer things that need to be run as root, the ability to run almost nothing SUID root, and text configuration files make it easy to lock down a system. Linux has been very virus-free, even when tasked with everyday dangerous chores like mail reading and browsing. This is a testament to basic good security design.

On the other hand, with Linux everyone is a programmer, and let's face it, not everyone knows secure coding. I don't always want to have to audit code when I install a package that is exposed to the Internet in some way. There are even some packages out there where the author explicitly states that the program is insecure and to not bother contacting him or her. This is unacceptable. The strengths of Linux security can be undone by one poorly coded application. But of course, that is true of closed-source systems too.

Mick Bauer (mick@visi.com) is a network security consultant for Upstream Solutions, Inc., based in Minneapolis, Minnesota. He is the author of the upcoming O'Reilly book Building Secure Servers With Linux, composer of the “Network Engineering Polka” and a proud parent (of children).

Archive Index Issue Table of Contents

Advanced search

Ultimate Machines

David A. Bandel

Issue #101, September 2002

What will the next generation think about our old servers and desktop clunckers?

Years ago, in the days before the Web was popular, the ultimate machine was equated universally with servers. These systems had the hottest, fastest CPU(s) one could afford, obscene amounts of memory (often as much as 64MB of RAM), huge, fast SCSI hard drives (usually two or three to spread the load) and were never commodity (Intel) systems—they were DECs, SUN Sparcs, IBM RS or HP UNIX servers. Back in those days, the big data centers belonged to Wang and others. Today, most server systems I install are or can be low-end systems with 1GHz processors, 128MB RAM, 18GB IDE hard drives that absolutely fly, at least compared to the clunky MFM or RLL drives of years gone by, and they're still overkill. Now, more than ever, you're likely to find the ultimate machine on the boss' desktop. After all, we can't have him or her waiting ten seconds for Outlook or Netscape to open, and how will he or she watch CNN while working in the bloated, monstrous word processor (with 95% of the “features” totally unknown to most users). Today's graphics cards have more RAM than the first disk drives I owned had storage space. And computing is only in its infancy. In a few years we'll look back on today and shake our heads, wondering how we ever got along with such slow, primitive systems.

CheckInstall asic-linux.com.mx/~izto/checkinstall

I find it hard to believe I've overlooked reviewing this particular package because I use it all the time. (All programs in this column are built from source.) This program is run instead of **make install** when installing packages from source. It builds (albeit crudely) RPMs, DEBs and TGZ (Slackware) packages. This will help control the cruft on your system as you install and remove source packages. I even use it on my Linux from Scratch systems (I install RPM and checkinstall early on). This is a must-have/must-use for *all* systems—production, test, whatever. Requires: bash, glibc.

CRM www.it-combine.com/crm

CRM allows you to track incidents (entities), assign them to folks for resolution, assign due dates, priorities and so on, then check up on all the activity. If you're running a service-oriented business, this particular application will be worth investigating. You even can set alarms on projects you don't want to extend past the due date. Easy to install and use. Requires: MySQL, Apache with PHP and MySQL, web browser.

LGeneral lgames.sourceforge.net

Many years ago I used to sit around with some friends for weekends at a time and play games like *NATO Division Commander*. I haven't done that in a long time, but this game brought back memories. I don't have the time now to sit around all weekend playing war games, and even if I did, my significant other would likely object. But I can play *LGames* anytime, even on sleepless nights, as long as I keep the volume down. Requires: libSDL_mixer, libSDL, libpthread, glibc, libm, libdl, libvorbisfile, libvorbis, libogg, libsmpeg, libartsc, libX11, libXext.

Crossword Generator www.ldc.usb.ve/~96-28234/crossword-0.8.tar.gz (download only)

If you like crossword puzzles, this program will provide you with all the puzzles you could want. You create the board and provide a list of words, and the program does the rest. What is needed is a tie-in to a thesaurus so the clue provides synonyms or definitions rather than the word itself. Documentation is provided in Spanish (as are dictionaries, etc.), but that's easily remedied. Requires: libstdc++, libm, glibc, TeX, LaTeX.

Find++ ios.free.fr/?page=projet&quoi=15

If there's one thing users like, it's simple, easy-to-use tools. But above all, they like graphical tools. The find++ utility will search your hard drive for words or phrases contained either in the filename or inside the file. Once a document is found, if the file type has been associated with a program, you can launch that program and open the file. It doesn't get much easier than this. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.

dnstracer www.mavetju.org/unix/general.php

Need to find out where a particular domain name entry is coming from? This will trace the authoritative information back to its source. The program has a lot of options for controlling how the query is run. Requires: glibc.

ippl (Internet Protocol Logger) pltplp.net/ippl

This month's pick from three years ago wasn't easy, as a number of good choices are still available, but ippl is probably the most useful. If you need to keep an eye on the types of traffic you have, ippl will do that well. It's somewhat improved since three years ago. Probably the best feature is that you can configure it easily to log only those protocols in which you're interested. Its drawback is a lack of support for other than the standard TCP, UDP, ICMP protocols, but few folks would need this anyway. Requires: libthread, glibc.

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

Grass Roots WiFi in London

Doc Searls

Issue #101, September 2002

Doc arrives in London to discover free, wireless net-access infrastructure being built by hackers using Linux and other handy materials.

Linux for Suits

The People's Infrastructure: Grass Roots WiFi in London

I travel around the US so much that I usually make sure I have local net-access numbers before I leave. Mostly this involves keeping an EarthLink account because EarthLink has a lot of numbers and requires no special client software to access them, but travel overseas is another matter. I've only gone twice in the last two years, and both times I departed unprepared.

The first trip was in the fall of 2000 to Lucerne in Switzerland. None of the local-access numbers for EarthLink worked, nor the numbers for my hosting ISP's local Swiss “partner”. I gave up and called numbers in the states.

The second trip was last week (it's mid-June 2002 as I write this). Before I left, EarthLink told me they were no longer (despite their name) offering overseas access, and I didn't have time to hassle with joining AT&T's system, which apparently has access numbers around the world but evidently exposes them only through their Windows client dialing software. Just before I left, I quickly engaged the services of two resellers of a “global roaming” provider called iPass (ipass.com) and headed for Munich. It looked easy: a $5.00 setup charge, plus somewhere between $.03 and $.23 per minute once I got on-line.

But I never could figure out how to get local numbers from either of the iPass OEMs (one eventually wrote to apologize for never getting me on the system, which could only be set up by phone call with a salesperson). However, it didn't matter because something miraculous arrived before I did: wireless net access.

Our hotel in Munich featured a wireless signal covering the lobby and meeting rooms. They sold it in the form of a card with an ID and a password hidden behind one of those scratch-off black patches they use on lottery tickets. You can buy two hours for 9 Euros or 24 hours for 29 Euros. (Not cheap, but very handy.) You simply fire up a browser, go to a URL specified on the card, enter your ID and password, and you're on for as long as the meter runs.

The event was JabberConf (www.jabberconf.com), where the techies on staff offered plenty of bandwidth as well—all of it free for anybody who wanted it. Unfortunately, their WiFi base station had some kind of problem, so everybody had to share an eight-port hub out in the hall. I wanted to be live on the Net in the meeting rooms, so I paid the hotel for the privilege, and it worked very well.

But the mindblower came when I arrived in London. After failing once again at dial-up from my hotel, I went down to the local internet café. There I paid a few pounds to sit for a couple hours with my laptop jacked into the Net, looking for local wireless access points. It took awhile, but I eventually hit pay dirt in the form of Consume.net (consume.net), the FAQ of which modestly describes its mission as “a collaborative strategy for the self provision of a broadband telecommunications infrastructure”. And indeed, that's what it is.

The top link on the left column of the main index page is “Nodes”. Clicking here brings up a map with little circles all over it, each bearing the name of an access point. Some, like Hyde Park and Greenwich University, are relatively obvious locations—if you're a local, which I'm not. Most are obscure: “twenteenth node”, “dude”, “neotokyo”. But after I subtracted out all but the operational nodes (marked green), I found that the nearest one, called Kynance Mews, was a short walk up Gloucester Road from my hotel on Cromwell, in South Kensington.

When I clicked on “get node info”, a page with an abundance of useful information appeared. In addition to the name of the node, it featured coordinates by both latitude/longitude and the Ordnance Survey, links to detailed street maps and aerial photographs and a list of every other node within 4,000 meters, including distance (to the meter) and compass bearing (to the degree). The description read, “Up and running—covering Kynance Mews and Kynance Place. This includes two cafés with good coffee and outside seating. Hurrah!”

So I walked straight up Gloucester, and sure enough, a signal presented itself as soon as I passed the parallel entries to Kynance Mews and Kynance Place. Down at the other end of the latter was a pretty little French café with outdoor seating called Petite Délice. So I went down there, ordered a coffee and a pastry, sat down, opened my laptop and found I was already on the Net. The node identified as Kynance Community Wireless, assigned me a DHCP address

and let me at the bandwidth. To further perfect the situation, the sky had turned to puffy blue clouds, the air was warm, the flowers blooming, the birds singing and South Kensington looked as postcard-perfect as you can imagine.

Two tables away another patron was sitting at a table talking about “access” to a tall and familiar-looking young gentleman walking a very friendly greyhound. I called over and said “Excuse me, is this your node I'm on?” “Yes”, he said, and came over. A look of recognition crossed his face and he said, “You're Doc Searls!” “Yes!” I replied. After uttering a delighted expletive, he held out his hand and introduced himself as Ben Hammersley.

That would be the same Ben Hammersley who writes for the *London Times*, the *Guardian* and O'Reilly Books, for which he's currently working on an authoritative piece on RSS. He also writes four different weblogs and had recently attended O'Reilly's Emerging Technologies Conference in Santa Clara, California, which is why I sort-of recognized him, because I was there too.

I shortly found out that Ben's node hangs off a Linux box and that Consume.net is served up by Linux and Apache as well. The connections, however, only began there. Soon we were joined by Ben's wife Anna (whose father is Olof Soderblom, inventor of Token Ring), and my assimilation into the Hammersley's techno-social network began, along with a crash course on grassroots infrastructure building that I'm sure would never proceed so quickly without Linux and allied free and open protocols—along with resourceful hackers to make the most of them.

For example, I learned about *warwalking*. The “war” in this case is not combat, but Wireless Access Reconnaissance. Matt Jones, architect of BBC Interactive, explained to me that warwalking was the pedestrian equivalent of wardriving, which reportedly got its name from the movie *War Games*. In either case, it's still about reconnaissance. Services like Consume.net just take some of the work out of it.

One of the other wireless patrons at Petite Délice told me there are downloadable scripts that will turn your laptop into something like a Geiger counter for WiFi. That way, you can walk around town with a closed laptop in hand, wearing headphones that make a sound when the laptop picks up WiFi signals. Matt even showed me hand-drawn schematics for his own variant of warwalking, called warchalking. With warchalking, hackers can use chalk to mark local-access status on curbs and sidewalks, much as service workers use spray paint on pavement to identify subterranean plumbing and electrical services. A closed circle might mean the presence of a closed access point, while an open circle (two halves, back to back) would identify an open network.

Matt plans to write this up by the time this hits print. You should be able to find pointers somewhere amidst his main personal site, www.blackbeltjones.com.

Not that everybody is going about this in perfect harmony. One of my new friends in London is Yoz Graheme (yoz.com), source of “Perl is Internet Yiddish” and other memorable lines. When I asked Yoz about Consume.net's Linux connections, he wrote back, “I think (they) are using Linux but being notoriously closed about their code....I'm not sure of the whole story there, but I know it's causing a fair degree of consternation.”

But Consume.net is not the only grassroots wireless movement in London. Another high-profile effort is free2air.org (www.free2air.org), which has a global scope with an apparently strong UK constituency. Naturally, their site is served by Apache on Linux.

As it happens, I was in London to address some high-level civil servants on how technologists and ordinary citizens in free markets were taking both infrastructure and government into their own hands—using Linux and other technology developments as examples. I spoke on my last day there, so I had a lot to report about what was happening locally, complete with digital photographs I had been taking since I arrived. One photo, titled “An Exploration of Infrastructure Irony”, was a telephoto shot of the top of an idle phone booth, with Ben Hammersley's base station in a window behind it. The entire time I spent on Kynance Place, I saw nobody using that telephone. Meanwhile, I have no idea how many people used Ben's node to access the Net for free—more than a few, I'm sure.

What these hackers were producing, I pointed out, demonstrated how the Net sees choke points as failures and routes around them. I also suggested that the same kind of thing would be happening with governments as well. “This isn't 'power to the people”', I said. “It's power from the people”, and there is a huge difference. The Net is made “of, by and for the people” in a very literal sense.

Much of the talk among my hacker friends in London was about politics. As with the DMCA (Digital Millennium Copyright Act) and subsequent lawmaking in the US, the UK has had its share of net-hostile legislation and organizations formed to fight it. Foremost among those is STAND (stand.org.uk). Matt Jones explains:

> STAND was created by a bunch of us in early 1998 to fight the Electronic Commerce bill, which we went several rounds with until it became the RIP bill, and then Act. However, in doing that, we created some faxing technology that we knew was useful on its own, and so created FaxYourMP as a notionally separate entity (FYMP is determinedly nonpartisan; STAND is

> totally single-issue lobbying—both happen to be run by the same people). STAND went into hibernation when RIP went through but was very recently (as in a couple of weeks ago) brought back in a new form by Danny O'Brien, who is one of the team, to fight the RIPA extensions. The FYMP code is a mess of PHP, MySQL and various other open-source things, and it mainly runs on FreeBSD boxes. We've often thought of opening the code, but we'd have to substantially clean it up first, and it's way too bitty. Besides, the value is in the setup as a whole, and it still takes regular administration and cash to keep it running. As I said, the software tools are free, but the things that cost are a) the fax calls, because local calls still cost money here and b) the data, which matches a user's postcode to an MP, which is copyrighted and has to be bought.

After I arrived back in the US, a flurry of e-mails delivered news that the STAND site described this way:

> As most of you will already have heard, the government has backed down from the RIP s22 Order that would have given access to traffic data to dozens of government departments. We thought you'd like to know that this U-turn was largely down to you.
>
> The FaxYourMP folk say that they relayed 1,789 faxes from last Monday and estimate that around 1,600 of those were related to the s22 RIP Order. That means that, on average, every MP received at least two messages expressing concern over the measure.
>
> We've received mail from constituents saying that their Member of Parliament called them directly to discuss the issue. We've had MPs mail us with advice. We've had TV companies and newspapers contact us after they'd been hassled by their readers and viewers. We've even had MPs writing letters to constituents explaining, mournfully, that there was nothing they could do—and then had their own voters explain to them how to attend Standing Committee debates and who to contact to help fight this order. Ah, those apathetic votees.

Power from the people. It happened with the Net. It happened with Linux. It's happening with WiFi. And it's going to happen with government too. Count on it.

Doc Searls is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Allocation of the Risks

Lawrence Rosen

Issue #101, September 2002

Make sure software licensors actually own all the copyrights they're making you license.

A license serves to allocate risks between the licensor and the licensee; it is an essential purpose of license warranty provisions. The BSD license, for example, contains the following clause:

> This Software is provided by the copyright holders and contributors “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services, loss of use, data, or profits, or business interruption), however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

(This clause was originally written in all capital letters to make sure every licensee reads it. I find all-caps text to be unreadable, so I rewrote it here in upper/lowercase.)

The important thing to notice is all risks under the BSD license are borne by the licensee. No matter how dreadful the software, no matter what damage it causes to your computer or your business, no matter what promises the licensor advertised about the usability or functionality of the software, the licensee accepts the software “AS IS” without any warranty.

All other open-source licenses I've read contain similar disclaimers. The licensee bears all the risks associated with using the software and creating derivative works from it.

I think the above is a fair allocation of risks, except for one risk that isn't expressly mentioned in the BSD license: the risk that the licensor isn't authorized to grant the license to the software in the first place. The BSD license (because of the “including, but not limited to” language) excludes this warranty of non-infringement.

Without a warranty of non-infringement, if the licensor doesn't actually own the copyright to the software he or she licenses or isn't acting under the authority of a license from someone who does own the copyright, the licensee, not the licensor, is accepting the risk of a lawsuit for copyright infringement when he or she accepts the software.

Here's how I corrected that problem in a new license I'm writing called the Academic Free License:

> Licensor warrants that the copyright in and to the Software is owned by the Licensor or is distributed by Licensor under a valid current license. Except as expressly stated in the immediately preceding sentence, the Software is provided by the Licensor, distributors and copyright owners “AS IS”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the Licensor, contributors or copyright owners be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the Software.

In the first sentence of that warranty clause, the Academic Free License allocates the risk of copyright infringement to the licensor rather than to the licensee. Between the licensor and the licensee, I'm convinced that the licensor is in a far better position to know whether he owns the copyright or has the software under a license that allows him to license it to others. The licensee, on the other hand, has no information on which to base a decision whether to accept the risk of copyright infringement. The Academic Free License, therefore, allocates that risk to the only party capable of determining the degree of risk.

For example, if the licensor knows he wrote the software himself, he owns the copyright. But if he merely copied the software from someone else or created the software while working as an employee of another company, he cannot honestly warrant against non-infringement.

Licensees may be unwilling to accept open-source software without some assurance from the licensor that they are not infringing. The absence of such assurances may inhibit the acceptability of open-source software. With a warranty of non-infringement, like the one I wrote for the Academic Free License, licensees can rely reasonably on the license language to protect themselves from accusations that they didn't care about who actually holds the copyright.

What would happen if the licensor gave that warranty of non-infringement but didn't actually own the copyright or didn't actually have a valid license to distribute the software? 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.

I'm interested in your thoughts about whether open-source licenses should include warranties of non-infringement of copyright. Please send your comments via e-mail to lrosen@rosenlaw.com. I'll report on the results of this informal survey in a later column.

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 Altos and Ukiah, California (www.rosenlaw.com). He also is executive director and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (www.opensource.org).

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #101, September 2002

Love That Aass

As a longtime subscriber of *Linux Journal* I have noticed the beer contest, and I have also noticed that Linux writers often make their own beers. I visited the USA in 1998 and 1999 because our son was then an employee of Intel in Portland, Oregon (now at RealNetworks in Seattle). In Portland, we visited the Brewers Festival, a great happening. My favorite USA beers were India Pale Ale from Bridgeport and Full Sail from Full Sail Brewing Company. We also visited the Full Sail Brewing Company in Hood River, Oregon.

Here in Norway, I prefer Norwegian beer from Aass brewery, and last year I discovered a new beer from this brewery, with Penguins on the aluminum can. Here in Norway we call these beer cans boxes (Norwegian: *boks*). My favourite beer therefore is a Linux box! So are my computers too.

With my Pine mail program and my ISDN internet connection, I will try to send you, as an attachment, my Nikon Coolpix 950 digital image of the Aass Ice beer can. The name Ice Beer and the Penguin logo comes from fermentation with

very low temperature I do not believe that the brewery has a specific interest in the penguin as a Linux logo, but I like both Linux and this excellent Ice Beer, so here are the penguins, for *Linux Journal*.

—Per Lillevold, Norway

Where Did YOU Go?

Not sure which version of SuSE you were using for your article [see Mick Bauer's "Staying Current without Going Insane" in the July 2002 issue of *LJ*], but with version 8.0, SuSE has changed things a bit. Best I can figure, YOU (YaST Online Update) will only check, recommend and update packages that fall into the category of security or critical updates. With older versions, YaST (the predecessor of YaST2) did have a mode that was capable of updating other packages. This has been removed as of SuSE 8.0, and there has been nothing that I'm aware of that was fixed to allow YOU to perform that function. There are a lot of users who have voiced their displeasure with this, and it is not clear if this was done intentionally, or if it was an oversight. I've been pretty pleased with SuSE since version 5.3, but I think there are a couple of weak areas with 8.0. I still give them the benefit of the doubt, though.

After coming to the realization that YOU wasn't gonna get things done for me, I found that there is another project (a re-port of apt adapted to support SuSE RPMs) afoot that is in fairly early stages, but appears to work pretty well. I'm still figuring it out, but it does allow me to keep KDE 3.0 up to date pretty easily. Apt4rpm works for SuSE versions 7.3 and 8.0. (See these related links: sourceforge.net/projects/apt4rpm and linux01.gwdg.de/apt4rpm/.)

—Kevin Vosburgh

Mick replies: Sad to say (?) I'm not running SuSE 8.0 yet. My SuSE systems are still on 7.1, so that was the version I covered in the article. Sorry for any confusion or inconvenience this may have caused you. Truth be told, I avoid "dot-0" releases because they tend to be, shall we say, "unripe". SuSE's "oversight" with regard to security vs. general updates in YaST2 is a case in point. (At least I hope it is. If it was a design decision, I would personally consider it to be a cynical one: as I noted in the article, stability can have security ramifications, and even when it doesn't, providing bug fixes regardless of security relevance is, or at least should be, an obligation of Linux packagers.) Anyhow, on behalf of both myself and Paranoid Penguin readers, thanks very much for the clarification and the tip about Apt4rpm!

Thanks Charles

Thank you, Charles Curley, for telling us about “Emacs: the Free Software IDE” [see *LJ*, June 2002]. With the limited print “real estate” you did a great job. I wanted you to be aware of how others have extended Emacs deep into the IDE world.

My first comment pertains to using the Emacs spell checker private dictionary. I was responsible for a Software Design Document on a military project. Because all our developers wrote code using Emacs, we adopted a standard abbreviations list, and after merging it with our vendor/military standards list we set it up as our common private dictionary. We used the Emacs spell checker to flag misspellings or nonstandard abbreviations or military/vendor terminology.

My other two comments relate to GDB. In software development, testing is important. I found the GDB user-defined functions with parameter-passing capability to be very powerful. I have literally created test verification documents of my software's results using Emacs and GDB. Also, in my line of work: real-time software and basic 2-D plots of data vs. time are always important. I wrote a simple Emacs macro to transform GDB output into a tabular file suitable as input to gnuplot. Thus, I get quality plotting of results while running my software via GDB within Emacs. Yes, many software development tasks are doable using Emacs.

—Harry Rockefeller

High-Performance Computing?

I noticed on page 92 (right column, top of page) of the June 2002 issue [see Richard Ferri's article “The OSCAR Revolution”]: “Workload management: Portable Batch Systems (PBS) from Veridian and Maui Scheduler (developed by Maui High Times Computing Center).” Many of my research colleagues have been closely involved with MHPCC, and I have spent some time there. We are all laughing our a**** off! I sent the following message to some of my research colleagues:

> In this month's *Linux Journal* (June 2002), there is an article on the OSCAR open-source cluster application effort. There is a bullet item list of tools and features on page 92, but this one really caught my eye: “Workload management: Portable Batch Systems (PBS) from Veridian and Maui Scheduler (developed by Maui High Times Computing Center).” That is not a typo in my e-mail. The article really calls it the “Maui HIGH TIMES Computing Center”. No doubt, the author intended to give new meaning to the phrase “smoking fast system performance”.

Since then, e-mails have been flying. The “suits” might not have much sense of humor, but we folks in the trenches love it. Keep up the good work.

—Todd Torgersen

Richard replies: That was completely unintentional on my part—I really thought I had read that name somewhere, and I thought it was a very laid-back Maui attitude. Of course, after the article came out in print, and I reread it, I realized my error, and I couldn't find any references to “Maui High Times” except for completely unrelated stuff, you know—I hope everyone maintains their sense of humor over my faux pas.

Archive Index Issue Table of Contents

Advanced search

Ultimate Is in the Eye of the BogoMip Counter

Richard Vernon

Issue #101, September 2002

Fast boxes, economic machines, living in CVS and world domination.

In this year's Ultimate Linux Box article, *LJ* Technical Editor Don Marti explains how you too can be the first on your block to build a machine that develops over 9,000 BogoMips. But now that machines with processor speeds of 1-2GHz, and even multiple processors, and gigabytes of RAM are quite common, building the Ultimate Linux Box isn't only about sticking the fastest and the biggest together (although that's still a lot of fun). Therefore, in addition to making recommendations on cards, motherboards, hard drives, etc., Don takes a look at some of the finer points of box building, such as box real estate, the advantages of building over buying and cooling. Though it's certainly a labor of love, Don has been working with vendors and others for many months now in order bring you building advice that has real value, whether you're building a computer from top-of-the-line components or one that represents a more modest budget.

Speaking of modest budgets, in Cooking with Linux this month, Marcel takes an alternative view of the idea of the Ultimate Linux Box, showing how you can obtain greater speed from humble resources by lightening the software load. He samples some lightweight software that includes a window manager with abundant features, a web browser and office software that manage to run all together in less than 32MB of memory.

Last month we ran an update to Charles Curley's November 2000 article on bare metal recovery. This month, Joey Hess shows how to avoid conscious backups all together by keeping not only your projects, but your entire home directory, in CVS. Joey admits the idea is a sure sign of an unbalanced mind, but that it also has many advantages, not the least of which is distributed backups.

Also in this issue, we have a report from John “maddog” Hall on his recent visit to the state of Rio Grande do Sul, Brazil to attend the Fórum Internacional de

Software Livre. Jon discovered that in Brazil they are taking the concept of world domination quite seriously, and the state of Rio Grande do Sul has had laws favoring the use of open-source software by government and business for some time now. His article points out a number of highly worthy free software projects.

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

Archive Index Issue Table of Contents

Advanced search

Going Embedded and Going Old School

Heather Mead

Issue #101, September 2002

Welcome, everyone, to a new monthly column that will highlight and point to articles, reviews, HOWTOs and other useful and fun features on the *Linux Journal* web site.

Welcome, everyone, to a new monthly column that will highlight and point to articles, reviews, HOWTOs and other useful and fun features on the *Linux Journal* web site.

New Embedded Category

We recently established a new Embedded article category for web articles that coincides with the new Embedded section that debuted in the August 2002 issue. The first article to post in this new web category is Guylhem Aznar's “Applications for the Sharp Zaurus”, which offers an extensive discussion about installing software on the Zaurus PDA. Guylhem also explains how you can sync data, flash the ROM and get on the Internet. For the full story, go to www.linuxjournal.com/article/5902.

Most-Read Articles

Although the DeCSS legal battle over DVD encryption left many people believing DVD playback applications for Linux were nonexistent, in “GNU/Linux DVD Player Review”, Jonathan Kent provides an overview of four applications for this purpose: Xine, VideoLAN Client, MPlayer and Ogle. In addition to DVD playback, some encrypted and some not, several of these applications are extendable to much more than DVD formats; a few plugins or downloaded libraries can get you far. Learn how at www.linuxjournal.com/article/5644.

On a more old-school note, Jim Hatridge undertook the project of adding his wife's “new” Compaq DeskPro 386/25e (to be fair, it is *new* compared to the 1987 XT it replaced) to their home network. The challenges were numerous—

she runs MS-DOS 5.0, but wanted only open-source software used—but using XFS and pcnfsd, some batch files and Samba, Jim was able to set up his machine as a fileserver to her machine. Then it was on to the next challenge—getting the 386 on the Internet. To see how that turned out and how you could set up something similar, go to www.linuxjournal.com/article/5837.

Most-Commented Articles

When it comes to putting articles on the web site, it's hard to gauge how readers will respond. Therein lies the beauty of the Comments section, where readers can post their opinions about the article's topic, debate some detail, point out something that was overlooked (or wrong) and add their own experiences. We were a bit surprised to see how Paul Barry's article “Perceptions of the Linux OS among Undergraduate System Administrators” took off with comments. Maybe we all felt a little dismayed that the next generation of programmers and system administrators were recycling the same “Linux is too hard to install” recitations. Take a look at what readers' responses were at www.linuxjournal.com/article/5650.

We get a lot of submissions for *Linux Journal*, and due to space limitations in print, many helpful tutorials, reviews and news items appear on the web site instead. Articles are available on the site dating back to 1994, and new ones are posted every day.

Heather Mead is associate editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Best of Tech Support

Various

Issue #101, September 2002

Our experts answer your technical questions.

Perl Won't Print without a Newline

When I run a Perl script (version 5.6.0) and try to print a number, for example, **print 5;**, the number will not be printed to the screen unless I include a newline character **print 5, `\n';**. If I change the shell to csh or ksh, the problem goes away. But if I set my shell to Bash or sh, the Perl script will not print the number to the screen, but it will print a string to the screen. My question is what do I need to change in the Bash shell to get it to work correctly with Perl?

—Blake Brezeale, blake.brezeale@bigfoot.com

You might have corrupted your Bash shell environment. In this case you can type **reset** at the shell prompt and press Enter.

—Usman S. Ansari, uansari@yahoo.com

Mixed-up Compiler Versions

I'm running Mandrake 8.1, which comes with gcc 2.96. I tried updating my compiler to 3.0.x using the Mandrake update software, and now I show parts of gcc 2.96 installed and parts of 3.0.x installed. As a result, I can't compile anything anymore.

—Rich Till, rtill@vetconnect.com

You should first uninstall all the gcc pieces (2.96 and 3.0.x), including libstdc and anything that is directly dependent on gcc 2.96. When all the pieces are removed from the system, re-install gcc 3.0.x and all the necessary pieces. Resolve any conflicts during the gcc 3.0.x install.

—Usman S. Ansari, uansari@yahoo.com

New Red Hat Install Won't Boot

I am trying to install Red Hat Linux 7.2 on an Intel hardware machine with a DPT VI (Adaptec) hardware RAID 5 setup. I go through the installation process okay and partitions are created. When I finish the installation process and press Enter to boot up, however, I get the error messages:

```
creating root device
mounting root filesystem
mount: error 19 mounting ext3
pivotroot: pivot_root (/sysroot, /sysroot/initrd)
     failed:2
freeing unused kernel memory: 220K freed
kernel panic: No init found  Try passing init=
     option to kernel
```

—Byron Rendar, brendar@pcc.edu

Ted Ts'o has written a solution for this problem at: www.redhat.com/mailing-lists/ext3-users/msg03575.html. You need to modify /etc/fstab; try mounting the root filesystem as ext2 if it cannot mount as ext3. Change the ext3 in /etc/fstab to ext3,ext2. (You can convert the ext2 filesystem to ext3 later.) A longer alternate solution also is available on the web site.

—Chad Robinson, crobinson@rfgonline.com, and Don Marti, info@linuxjournal.com

Getting On-Line with Cable

How do I connect to the Internet using digital cable? My provider is Cox in Oklahoma City, and they do not support Linux machines. My NIC is a Realtek 8139 10/100. SuSE Linux detects the card, and it's enabled with DHCP. During bootup my card is detected but no IP address is assigned.

—Matt Reynolds, mattreynolds@cox.net

Is your provider actually using the DHCP protocol? Some providers are switching to PPP over Ethernet, which is not compatible. Check with your local Cox office to be sure; Cox uses both, depending on where you are in the country. SuSE supports PPP over Ethernet, but it's documented under ADSL or T-DSL in the manual, and the YaST2 configuration screen is called “DSL configuration”. Install the pppoed package, and see sdb.suse.de/sdb/en/html/hoe_adsl_pppoe.html.

—Chad Robinson, crobinson@rfgonline.com, and Don Marti, info@linuxjournal.com

Need to Set High-Numbered DHCP Options

As I work with IP-telephony (Nortel), I need to use vendor-specific codes in my DHCP server. Is there any DHCP server out there for Linux where it is possible to use vendor/option codes higher than those the Red Hat distribution uses?

—Bjoern Arstad, chancho@online.no

If your DHCP option code isn't supported by name, you can include it in dhcpd.conf with:

```
option option-nnn 'value';
```

where nnn is the option code as a three-digit decimal number. See man dhcp-options-dhcpd.

—Don Marti, info@linuxjournal.com

Ethernet Masked Ball

I would like to be able to modify the MAC ID of my Ethernet card in much the same way I can with my Linksys router. Is there an easy way to do this?

—Mike O'Doherty, mgi1356@motorola.com

You didn't specify which Ethernet card you have; some allow this, while others do not. If your card does allow this, you can use ifconfig as follows:

```
ifconfig eth0 hw ether 001122334455
```

For those using cable or DSL modems, note that this is a useful trick if you have set up your service using a Windows box and want to install a Linux firewall/gateway. Most providers track the MAC address of the workstation's Ethernet card and can be hard to deal with if this changes. With the above command, you can force your Linux gateway to have the same MAC address as your original client system.

—Chad Robinson, crobinson@rfgonline.com

Touch Me, It's So Easy to Leave Me…

What is the required XF86Config-4 setup to use a Dynapro (3M) touchscreen? Do I need to enable it with something similar to **xsetpointer NFI3** after X starts?

—Shane Kennedy, skenn@indigo.ie

You can find updated drivers for 3M (formerly Dynapro) touchscreens at www.cdp1802.org/mmmtouch.

—Robert Connoy, rconnoy@penguincomputing.com

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #101, September 2002

Monthly new products column.

Storix Backup Administrator

Storix Backup Administrator (SBA) version 4.0 is available for system backup and recovery of most distributions and for x86- and PPC-based systems. Using a graphical interface, SBA provides backup management of standalone machines or entire networks. Options are available for backup scheduling, performance statistic reporting, overwrite and retention policies and tape striping, among others features. SBA can restore data on the same or a different hardware configuration, and you can recustomize your system by changing filesystem types, adding software RAID devices and converting to LVM partitions. All major filesystem LVMs and software RAID devices are supported.

Contact Storix Software, 3707 Fifth Avenue, Suite #125, San Diego, California 92103, 877-786-7491 (toll-free), www.storix.com.

CodeTEST for Embedded Linux

Applied Microsystems entered the embedded Linux market with the release of CodeTEST for embedded Linux, a software test and analysis tool suite. CodeTEST enables developers to conduct comprehensive performance testing, memory testing, RTOS and source-code level execution trace, hardware tracing of kernel and driver code and code coverage on applications based on embedded Linux. CodeTEST can measure more than 128,000 real-time function executions at once, looking for errors in algorithms, call-pair distribution and service routines, and measures CPU consumed by functions and tasks. CodeTEST can be purchased as a complete suite of separate modules for performance, memory, trace coverage and advanced coverage.

Contact Applied Microsystems, PO Box 97002, Redmond, Washington 98073, 800-426-3925 (toll-free), info@amc.com, www.amc.com.

Rackable Systems 1U 1800

Xeon processors will power Rackable Systems' back-to-back 1U 1800 model server. Rackable's 44U cabinets hold up to 88 1U servers and up to 176 Xeon processors. Rackable also will be using the following new Intel products in their servers: the SE7500CW2 server board, the SE7500WV2 server board and the SRSH4 server platform. The new server can be used in Rackable's patent-pending chassis, designed to allow for installation into both front and back of a standard 19" wide, 30" deep cabinet or both sides of a two-post “Telco-style” rack. The 1800 server comes with 8GB of memory, one to four HDDs, one PCI X expansion port and one or two 10/100/1000 Ethernet connections.

Contact Rackable Systems, 721 Charcot Avenue, San Jose, California 95131, 408-321-0290, sales@rackable.com, www.rackable.com.

GNOME 2.0

The GNOME Foundation released version 2.0 of the GNOME Desktop and Developer Platform, which features a faster and more powerful Nautilus file manager and dozens of new utilities and applications. A simplified configuration process also is in place for version 2.0. Included in this release are an enhanced GTK 2.0 toolkit, libxml2, add-ons for Glade, Python and CORBA, and many new libraries and widgets. Other improvements in the new version include anti-aliasing for fonts; a dynamic, centralized configuration system; rewritten terminal application with tabs and profiles; and a lightweight help application called Yelp, among many others.

Contact The GNOME Foundation, www.gnome.org.

PCI-8213 and PCI-8214

The PCI-8213 and PCI-8214 are two- and four-port, 64-bit fast Ethernet cards with bypass capability that are suitable for use in firewalls, network traffic monitoring, on-line gaming, high-availability and Internet-dependent applications. With bypass capabilities, during crashes and downtime the onboard relays crosslink channels 1 and 2. Fail-safe data transmission is achieved by linking inbound and outbound network traffic, even when the hardware or software fails. Both boards run on a 64-bit, 33/66MHz PCI bus but also are compatible with 32-bit PCI slots. Each Ethernet port has two self-diagnostic LED displays to show link and 10/100 status.

Contact ADLINK Technology America, Inc., 15279 Alton Parkway, Suite 400, Irvine, California 92618, 949-727-2077, usa@adlinktech.com, www.adlink.com.

Fax Messaging Server

Faximum Software has released Fax Messaging Server (FMS) version 2. Running on Linux, FMS 2 integrates with existing e-mail servers and enables Linux, Mac and Windows machines to send and receive faxes using the same tool as e-mail. FMS 2 allows users to combine e-mail addresses and fax numbers in the same message or e-mail group, delivers faxes to users in the same inbox as their e-mail, can bypass the long-distance phone system and is administered via the Web. Compatible with most distributions and most SMTP e-mail servers, FMS 2 has been designed to work with Caldera's Volution Messaging Server and the SuSE eMail Server III.

Contact Faximum Software Inc., 1497 Marine Drive, Suite 300, West Vancouver, British Columbia, Canada V7T 1B8, 604-925-3600, sales@faximum.com, www.faximum.com.

Arkeia Virtual Server

Arkeia Corporation introduced the Virtual Server, which provides local and remote data protection services adapted especially to ISPs, telecom specialists and cable operators. Two backup options are available: backup of customers' servers hosted at ISPs' locations or remote backup of servers at customers' locations to a centralized backup server hosted by the ISP. Arkeia Virtual Server offers simultaneous site hosting, multistreaming from one host for large volumes of data, client multiplexing, data transfers over limited bandwidth and library sharing between hosts. Privacy is protected through dedicated access ports, individual catalogs of backed up data, password and proxy authorization and no shared backup mediums.

Contact Arkeia Corporation, 1901 Camino Vida Roble, Suite 200, Carlsbad, California 92008, 760-602-8590, sales@arkeia.com, www.arkeia.com.

Archive Index Issue Table of Contents

Advanced search