Linux Journal Issue #119/March 2004

Features

Indepth

Embedded

Toolbox

Columns

Reviews

Departments

Delivering Effective Presentations with OpenOffice.org's Impress

Rob Reilly

Issue #119, March 2004

Choosing the right presentation software is only the beginning of preparing an effective talk. Here's how an experienced tech speaker uses OpenOffice.org to get the message across to the audience.

As a Linux techno-guru, you recently have been volunteered to give a talk at a high-profile industry conference. Your stomach starts to churn as you realize that you might have to learn Microsoft PowerPoint. Oh, the humiliation! Fear not, OpenOffice.org Impress runs on your Linux box. Think how calm you'll be in front of all those people knowing that your trusted old friend Linux is right there at the other end of your wireless mouse.

It's All Showbiz

Technical presentations are about transferring knowledge to the audience. The content is the show, you are the showman and the presentation technologies are the showbiz tools. Laptops, graphics, projectors, lights and presentation software all serve to organize the show and focus audience attention on the content.

As a techie myself, I know that technology is a siren call to be explored and fiddled with. We have to resist the temptation to tweak for now, though; we have a show to put on. Here, I discuss the basics of building a slideshow with OpenOffice.org Impress and a few ways to get your show organized. Then, I throw in a few power tips, so you will look like a pro in front of your adoring audience.

What Is Impress?

The OpenOffice.org Impress package lets you quickly construct and deliver an electronic slide presentation. With it, you can add graphics, make handouts and convert your slides to Web pages. It can do all the jobs needed to put on a great presentation. You simply add your content and personal style.

Loading OpenOffice.org Impress onto your Linux machine is simple. You might have OpenOffice.org already installed. Otherwise, go to www.openoffice.org and download the latest version; version 1.1 is around 77MB. The .gz file can be unzipped and put in its own directory. Then, run the setup file, fill in the blanks and you're ready to go.

You don't need a 3.0GHz laptop to run Impress. It runs fine on old 166MHz Pentium desktops with an 8MB video card and KDE or FVWM2 X window managers.

Quick Presentation Creation

Fire up OpenOffice.org Impress and take a spin through the program. Let's first look at some basic functions and then come back and get our content organized. The easiest way to start a new presentation is with the AutoPilot feature, which provides a basic beginning framework. Graphics, text, animation and formatting can be added after you enter your content.

To build a presentation, click File→New→Presentation. In the next window, select Empty Presentation and then click Next. This next window lets you choose backgrounds. Click Next for a window that lets you choose slide transitions. Our first presentation doesn't have any transition effects and is controlled manually by the Page Up/Page Down keys or the mouse, so click Create. The Insert Slide window should appear here. It lets you select the slide layout. For our first sample, use the layout style named Title, Graphic, Text. Highlight that layout, click OK and watch your first slide appear (Figure 1).

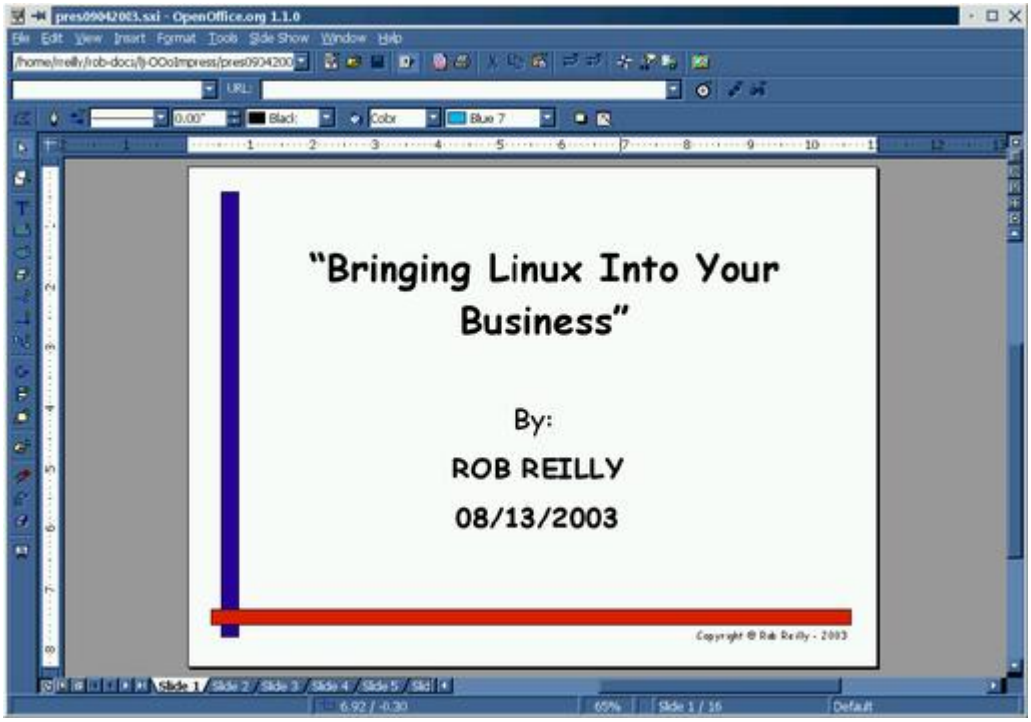


Figure 1. Building a presentation starts with the first slide. Here is the main Impress window.

This first slide offers you a place to enter your slide title, a picture on the left and your bullet points on the right. Using the default font size, the bullet point area forces you to work with only four or five lines. That's good because many experienced presenters put too much information on a slide. Use the slides and bullet points for a prompt and tell the story in your own words. By doing so, your audience sees you as an expert who thoroughly knows the subject.

The exercise of creating the first slide should take about 5–10 minutes. I intentionally went through the entire process because I wanted to get you right into the Impress program. Knowing how to build a basic slide provides you with the understanding necessary to organize your talk and build subsequent slides.

Forget about outlining your talk on paper; use Impress itself. Brainstorm your ideas logically, using the titles as main topics, each topic being a new slide, and put three or four bullet points under each topic for the details. You can rearrange and edit them after you've laid out all the material.

Adding Slides, Text and Graphics

Adding a slide is easy. Start by clicking Insert→Slide. Select the type of slide and then click OK. Your slide shows up on the screen, and you can enter text or whatever you want. To duplicate the last slide, click Insert→Duplicate Slide. Now you're ready to add some text. Click the Text icon on the upper-left edge of the main Impress design window. Move the cursor to the slide and left-click a location. Now, type in your text. You can move the text to another location by

dragging the middle of the text box. To resize the text, highlight and choose a new size in the point size space, at the top of the design window.

Adding graphics is equally as easy. Click Insert→Graphics, and select the graphic file to use. Click OK and the file is pulled into the slide. Resize the graphic by grabbing a side and moving the edge. To move the graphic, grab it in the middle and reposition to the desired location.

Making Slide Handouts and Notes

Handouts are groups of slides in a book form, with four slides per page by default (Figure 2). When printed, they can be stapled together or put in a binder and given to audience members before the show. For technical presentations, the handout allows the audience to take notes on a picture of the slide being discussed.

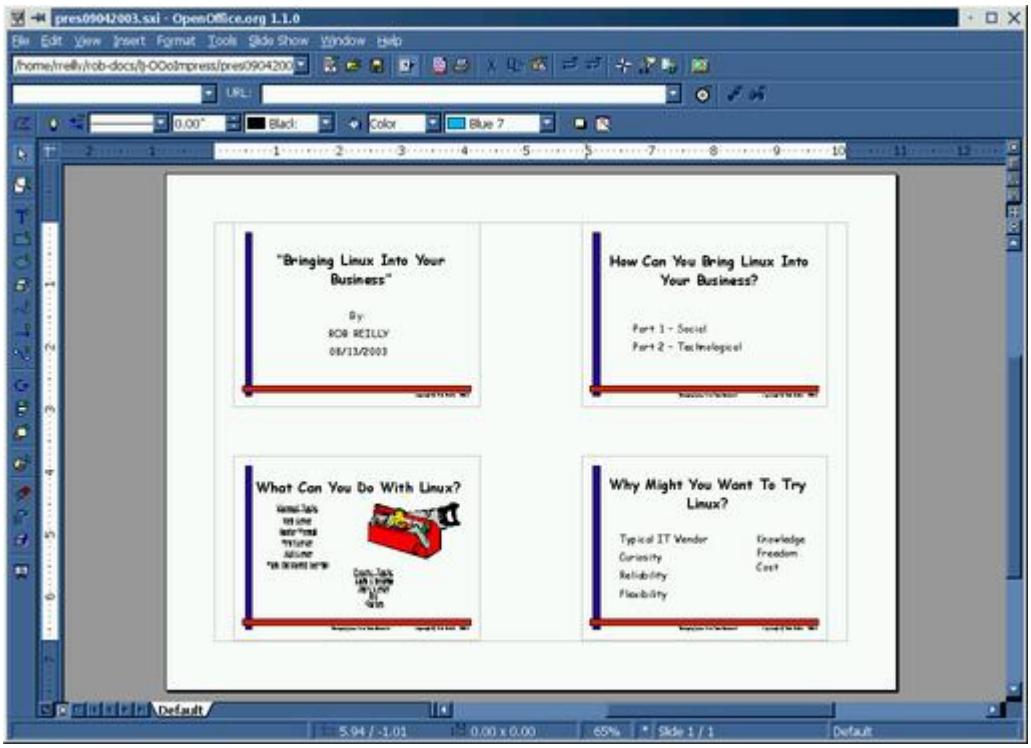


Figure 2. Previewing Handouts, Four per Page

To create handouts, switch to Handouts Mode by clicking the Handouts View icon along the upper right-hand window slidebar. Print the handouts by clicking File→Print→OK. Your entire presentation is printed with four slides per page.

Presentation notes add information that you or the audience find useful, but they don't show up when you run the slideshow. Print off a copy of the notes on paper if you have a big presentation and need some additional prompting.

To add a note to a slide, switch to Notes mode by clicking the Note View icon along the upper right-hand window slidebar. Select the Click to add notes

section at the bottom, and enter any information you like. The slide then is displayed in the top part of the view (Figure 3). Finally, click the Slide View icon (above the Note View icon) to return to your slide, minus the notes.

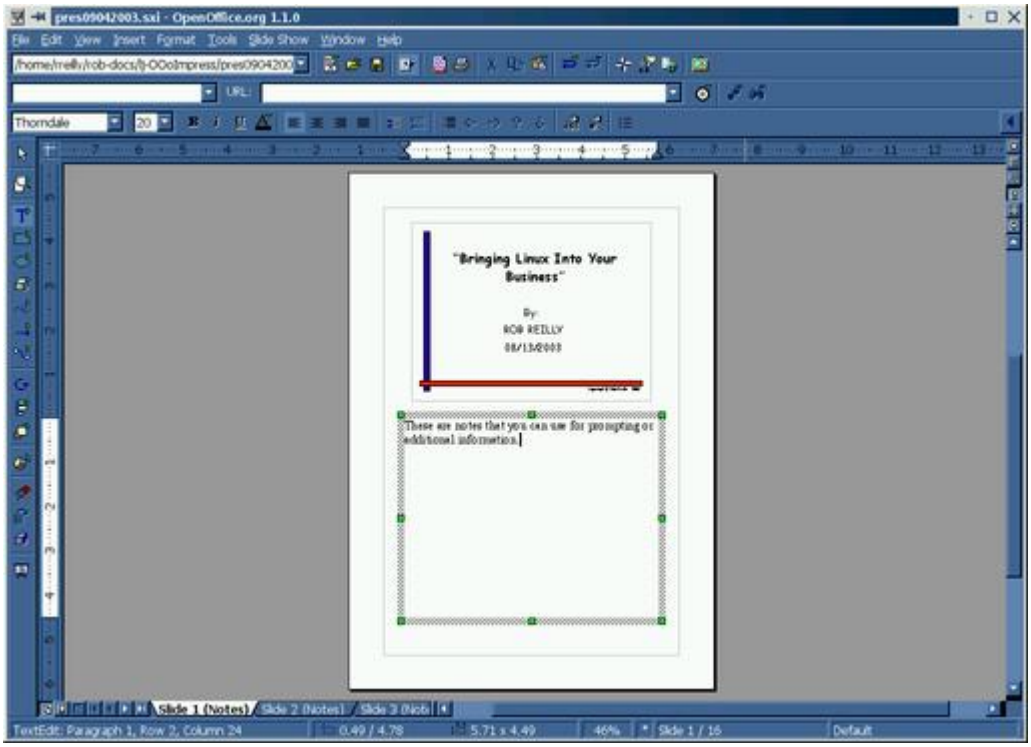


Figure 3. Adding Speaker Notes in the Notes View Screen

Running the Presentation

Naturally, you'll want to rehearse your masterpiece, once you've finished it. You always should practice your presentation before the main event. Switch to full-screen mode and page through your presentation with the Page Up/Page Down keys or the mouse buttons. Click the Start Slide Show icon on the right side of the windows slidebar (below the Note View icon) to get into Slide Show mode. Press the Esc button to return to the Impress Drawing View mode.

Converting Slides to Web Pages

Creating and delivering your presentation to your audience has been our focus thus far. But, why not provide the slides to your audience on a Web page (for review later) and look like a real professional speaker? By directing your audience to your Web pages, you also are able to show them your company information, your bio, other presentations and whitepapers and Web articles. Nothing makes a speaker more credible than providing valuable information and services to his or her audience, even after they leave the talk. That little extra effort of putting your presentations on your Web site can set you apart from the small-time tech speaker.

OpenOffice.org Impress makes it easy to set up simple Web pages based on your slides. Begin by opening your presentation in Impress, and click File→Export. Select the working directory for the HTML and graphics files. Next, enter the filename of your main HTML page (without the .html extension) and click Export. On the Assign Design screen, select Next. On the Publication Type screen, select Next. On the next screen you can save graphics as JPEGs, so select Resolution medium—600×800. Fill in the author, e-mail, home page and information on the Information On The Title Page screen. Then, select Link to Original Presentation, and click Next. Choose the types of buttons you want for navigation around your pages and click Next. On the Color Selection screen, select Create. Finally, you can save the HTML design you created with a name on the Name HTML design screen. Save it and you're done.

You now have a cool set of basic Web pages that showcase your slides. The title page carries your name, e-mail, Web site URL and additional information (Figure 4). There's also a link to the original Impress presentation.

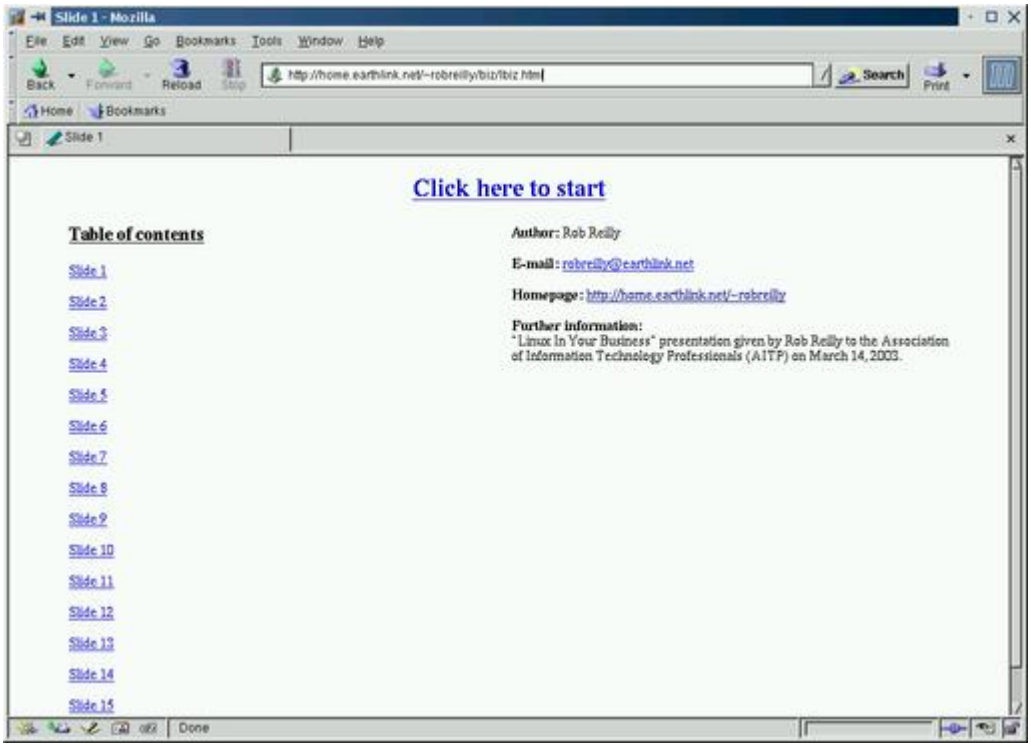


Figure 4. The main Web title page includes links to other relevant information.

To look at your handiwork, go to the working directory and open the HTML file that you named when you first exported your presentation (Slide 1 example—see Figure 5).

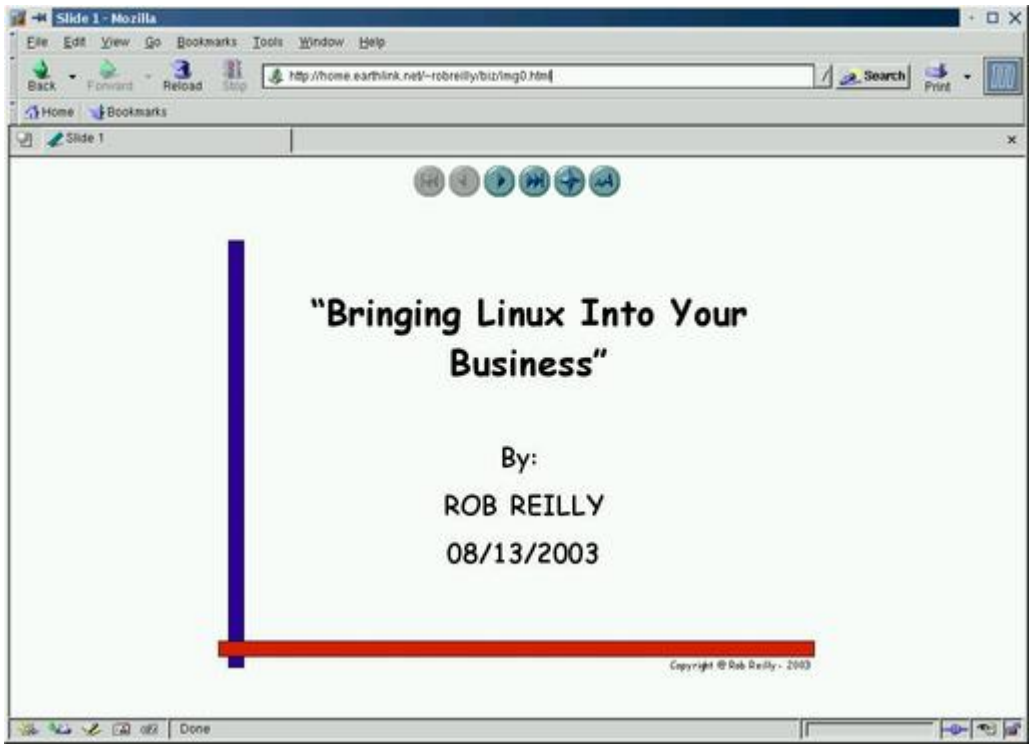


Figure 5. Previewing the Web Version in a Browser

Once you're satisfied with the look of the Web pages, upload the whole directory—don't forget your original presentation or .sxi file—to your main Web site. Add a descriptive link to it from your main page and then give the URL to your audience. Check the presentation pages on your Web site after uploading, of course, to make sure everything is there and correct.

Your Time in the Spotlight

There are two pre-presentation tasks that you must do to ensure a great presentation. You are going to be in front of people because you are the subject-matter expert, but relax and have fun as an entertainer—make it enjoyable. Second, end your presentation with a call for action, something like “Now go try Impress”, instead of simply saying “thank you” and smiling at all that thunderous applause.

Unless you are an extremely experienced speaker, do a full rehearsal at least twice before doing your live talk. By rehearsal, I mean set up a room, your laptop, the projector and go through the whole presentation in real time. Don't be afraid to use a stopwatch, as you need to be absolutely sure you can fit your presentation into the allotted time. You probably will have some editing to do. The best idea is to start putting your show together as soon as you get the assignment. Nothing messes up a show as much as going overtime or not covering all your material. Plan for contingencies; for example, if the previous speaker went long, what will you do?

Also, arrive early at your venue to set up the equipment and room the way you want them, and ask the event staff to help you. Test your laptop with the projector at the venue before the presentation. Finally, pack everything you think you might need, including transparencies and extension cords, just in case.

Wrap Up

Technical presentations can be fun and profitable. Use Impress on your Linux laptop and enjoy being in the spotlight. Plan ahead, make sure you rehearse enough and stay on time. Good luck!

Rob Reilly (robreilly@earthlink.net) is a technology writer and speaker whose articles appear in LinuxToday.com and *PC Update* magazine. His “Impress-ive Presentations” seminar covers OpenOffice.org Impress and technical speaking in greater detail. Visit his Web site at home.earthlink.net/~robreilly.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Eleven Tips for Moving to OpenOffice.org

Bruce Byfield

Issue #119, March 2004

Replacing a complicated piece of software like an office suite can be a major undertaking. But, if you apply a few simple rules to make sure your needs are met, you'll be ready to be productive on the new software right away.

For the past 18 months, I've been badgering people to try OpenOffice.org (OOo). Slowly, I've come to realize that talking about free software isn't enough. The problem isn't that people don't like the idea of free downloads or of joining a project and having a voice in its development. After all, what's not to like there? The problem is that being sold on free software rarely is enough to guarantee a smooth transition to OpenOffice.org.

What follows are suggestions I've formulated for making the switch. Even the most open-minded people usually have assumptions to discard. They still have preparations to make and features to try. Most of all, they have some hours to log before they really can decide whether OpenOffice.org is right for them. Nobody can help an individual or an office switch if curiosity or a willingness to explore is missing, but if you pay attention to the tips that follow, you should be able to remove much of the pain from the process.

1. Don't Expect Features to Be Missing

People always say, “I'd love to use OpenOffice, but...”, and then they name a feature they can't live without. If they've glanced at OpenOffice.org, they might claim their must-have feature isn't there. Most of the time, I can respond by telling them where to find the feature, and they fall into an embarrassed silence and change the conversation.

Sometimes, this missing must-have response is sincere, but more often it's an excuse. Either way, I suspect, the assumption behind it is that free software always is inferior to its proprietary equivalent. When a casual search doesn't immediately unearth the feature, this prejudice is reinforced.

Even if the assumption were true, and Linux, Apache, GIMP and Mozilla users all know that it isn't, the assumption wouldn't be true for OpenOffice.org. Although at the time of this writing OOo is at version 1.1, StarOffice, its Sun-owned predecessor, has a longer history than people think. In fact, the first version of StarOffice was a DOS word processor released in 1985. With two decades of development behind it, the OpenOffice.org code is mature and mostly complete.

Admittedly, other word processors do have features OpenOffice.org lacks. It doesn't have MS Word's grammar checker, WordPerfect's Reveal Codes feature or FrameMaker's master pages. But, OOo has features these rivals don't.

The point is, there's no need for a pessimistic view of OOo's features. Usually, you can be optimistic and assume the feature is somewhere in the menus. It may not be in quite the same form with which you are familiar—for example, OOo's outlining tool functions quite differently from MS Word's—but odds are you can find it in some form, however mutated.

2. Don't Expect Features to Be in the Same Place

No doubt about it, the OOo interface is similar to MS Office's. The general menu structure often is identical, down to the confusing placement of Configure and Options in the same menu.

This similarity between interfaces can ease the switch to OOo, but it can be misleading, too. In places, OOo has cleaned up and rationalized the MS Office menu structure. Tables, for example, do not rate a separate menu in OOo; they are placed in the Insert menu instead. At other times, the same feature has a different name: MS Word's Autosummary, for example, is AutoAbstract in OOo.

In other words, OOo is a mixture of the familiar and the new. Fortunately, basic functions usually are in their familiar places, so unsophisticated users are unlikely to get lost. However, if you're an advanced user, you may need to be more flexible. If a tool isn't where you expect it, think about what other menu it might be under or what else it might be called. If your imagination fails, look at the MS Office Feature comparison in the Help files, or look at my more detailed comparison at www.raycomm.com/techwhirl/magazine/technical/openofficewriter.html. In most cases, you should find what you need.

3. Don't Expect to Need Training

Even though you may stumble over the placement of some tools, you probably don't need a long transition period before you or your company can use OOo productively. Most likely, the transition can be completed in well under a week. Basic users can make the switch easily because they use a word processor as

though it were a typewriter. If they want to italicize a word, they don't use the Emphasis character style. Instead, they highlight the word and click the italic icon. If they decide they would rather use a bold font to emphasize words, they go through their document and change the formatting on each word separately.

The basic methods are not efficient ways to use any word processor, let alone OOo. But, people who work in this way use only a small set of tools. In OOo, these features generally are where such users expect to find them. Character and paragraph characteristics, for instance, are found in the Format menu or on the toolbar, and the spell checker is in the Tools menu. The transition to OOo may be an ideal opportunity to learn more, but meanwhile, users can complete their daily work with almost no interruption.

Advanced users may take a day or two longer to adjust. However, in the same way that knowing one language can help you to learn another one from the same region, knowing one word processor helps advanced users learn a different one. Advanced users know what to expect, and they typically have the confidence to search for it on their own. As a result, advanced users shouldn't need to be trained on OOo, either—they can train themselves.

4. Don't Rely on Import/Export Filters for Exchanging Files

At first glance, OpenOffice.org looks to be ideal for exchanging documents with other office suites, especially MS Office. Several MS formats are available when saving a file, as is a batch converter (File→AutoPilot→Document Converter). Moreover, in Tools→Option you can set OOo to save to MS formats by default and to preserve the VB scripts it can't use. What more can you need? Patience, for one. Truckloads of spare time, for another.

The truth is, there never has been a completely reliable import or export filter in any office suite. Chances are there never will be. If there is, my money is on cross-compatibility between OOo, KOffice and/or GNOME office. These formats are all open source, so at least the development time will be shorter. But, even with open-source formats, filters are going to cause problems for the immediate future.

Why? For one thing, writing filters is an intensive and unglamorous job. For proprietary companies, making the perfect filter is too expensive—aside from the fact that they don't want you using rivals' software. For free software developers, more interesting projects always are available. Besides, the people who need filters mostly are not developers, so developers are less likely to see the need for them.

Just as importantly, many filters involve proprietary formats. This means developers need to do reverse engineering, a difficult, time-consuming and sometimes legally risky process. Filters for MS Office, the main concern, are especially difficult because the format often changes and may not be even backwardly compatible with earlier versions of itself.

That said, OOo's native XML format makes writing filters easier, and its MS Office filters are among the best I've seen. Yet even these filters are far from perfect, and users who rely on them should reconcile themselves to a regular dose of manual reformatting.

If you insist on using other office suites with OOo, try to limit the editable documents exchanged to short, simply formatted documents. Search OOo Help for “About Converting Microsoft Office Documents” to see a list of elements you should avoid. For those elements you do use, you can improve the results if you use only styles and ensure that both office suites have access to the same fonts. Even then, you can expect anything other than the simplest bullets to be garbled. You might consider making a list of allowable formatting to minimize problems.

If users of other office suites don't need to edit a document, select File→Export as PDF and send it as a PDF file. PDF files are close to an open standard; therefore, this is one filter on which you can rely.

The best format solutions for exchanging editable documents between office suites probably are HTML or Simplified DocBook. Both can be viewed in modern browsers, and they can be opened as text files in word processors, if nothing else. Better yet, get your company or community using OpenOffice.org by itself. You still may need to share documents with outsiders, but your daily life will be simpler.

5. Make a List of How to Do the Basic Functions That You Need

Before you switch to OOo, make a list of the basic tasks you or your division does in an office suite. Try to keep the list to less than 20 tasks. Then spend half an hour experimenting or browsing the Help section of OOo. Write down how to do these tasks on file cards and distribute them to everyone. As each person becomes comfortable with the basic tasks, replace the first file cards with instructions for less common tasks. In days, or even hours, you should find that no one needs to use the cards.

6. Use the Available Help

OOo comes with a fully developed help system. In earlier versions, the help files often lacked context and gave circular definitions of features. As of version 1.1, though, the Help section actually has become an asset instead of a formality.

Sometime early in the transition, have everyone read the first four links on the Welcome to the OpenOffice.org Writer Help page. The links provide a good overview for delving deeper into the program. You also should consider suppressing your natural annoyance and enable the startup tips and office assistant for a few weeks. Both of these features offer useful information in small chunks. Although your understanding may seem fragmented at first, in the long run the tips are a painless way to learn.

7. Start with the AutoPilot Features

One of OOo's features for newcomers is a series of wizards that lead you through the process of setting up basic documents, such as a letter or memo. You may not find the results exactly fit your needs, but they are a quick way to get started with OOo. Look under Files→AutoPilot. Just as importantly, try comparing the instructions in the AutoPilot with the final results. It's a good way of knowing what office suites in general and OOo in particular can do.

8. Learn to Use Styles

If you're the type of user who manually applies formatting, mark your switch to OOo by learning how to use styles. Styles save you time in any word processor by allowing you to make formatting changes once and have them ripple through the document. Styles are especially important in OOo, because they give you templates not only for paragraphs and individual characters but also for pages, text frames and lists. Go with this flow and you not only minimize difficulties, you increase your efficiencies.

The key to styles in OOo is the Stylist, a floating palette found at Format→Stylist. You can use the Stylist to apply styles quickly as you type and to modify existing styles or create new ones. It lists styles using several different filters, so you can locate the ones you need quickly.

9. Learn to Use the Navigator

The Navigator (Edit→Navigator) is another floating palette. Like the Stylist palette, it is a key feature in using OOo effectively. As the name suggests, one of the functions of the Navigator is to help you move quickly to different parts of the document. Tables, OLE objects or pages—you can jump to almost any element in the document you want. Elements are numbered as you create

them, but if you also give them descriptive titles, the Navigator can display them, making jumping around even easier.

Don't let the name mislead you, though. The Navigator is far more than a map of your document. Switch to Headings and it becomes an outlining tool, with the ability to move entire sections and raise or lower the level of headings with the drag of the mouse. Open a Master Document, and it becomes a table of contents. You even can use the Navigator to add a Reminder to the text.

In short, you probably will come to spend a lot of time with the Navigator. And, it is something for which your experiences with other office suites doesn't prepare you.

As an aside, the default size of the Navigator may be too small. Drag its sides until the Navigator is at least half again as large as the default, and you can use it without eyestrain.

10. Look for Hidden Functionality

Unexpected features or shortcuts can be found in any software. These aren't quite Easter eggs but half-hidden functions that rarely are emphasized or mentioned in the Help. For example, I quickly found Edit→Undo. Because I generally use the menus or keyboard, though, it took me several weeks to realize that if I selected the Undo button on the taskbar, I could choose the exact level of Undo to which I wanted to revert. Similarly, if I want to insert text automatically each time I use a style, I can use the Before field on the Options tab for list styles and attach that list style to a paragraph style. Then, every time I use that style, the text in the Before field appears without me having to type it.

Such surprises do three things: they give you confidence in your knowledge of the program, they encourage you to keep learning and they offer shortcuts for your daily work. They're well worth seeking.

11. Take Time before Making a Decision

The first few times you start OOo, your impression simply may be that it's new. It doesn't look the same as your old word processor, it isn't arranged in the same way and it does some things differently. For some people, the newness alone is enough to make them cut the experiment short.

Instead of jumping to conclusions, however, wait and learn the program before making a decision about OOo or any of its features. Forget about your sense of being overwhelmed with the new and try to get on with your daily tasks. Spend at least 10–15 hours doing routine work before you even start to make a decision. Then sit down and list the pros and cons of using OOo. If you decide

against OOo, keep it in mind and try another version in a year or two. In the future, you may find that it fits your needs better. If you're a decision-maker at a company, you also might consider contacting the OOo community to see whether your company could sponsor the development of the features you need. If you do decide to keep OOo, congratulations. You did your preparation, and you're making the right choice.

Bruce Byfield was product manager at Stormix Technologies and marketing and communications director at Progeny Linux System. He also was a contributing editor at *Maximum Linux* and the original writer of the Desktop Debian manual. Away from his computer, he listens to punk-folk music, raises parrots and runs long, painful distances of his own free will.

Archive Index Issue Table of Contents

Advanced search

Renaissance—A Cross-Platform Development Tool for Linux and Mac OS X

Ludovic Marcotte

Issue #119, March 2004

Prepare to move to a Linux desktop by writing your apps with this flexible framework now.

Renaissance is a free development framework, licensed under the terms of the GNU LGPL, used for easily creating portable user interfaces. It allows the developer to create rich user interfaces by using an open, simple and standard format, XML.

When not using Renaissance, Objective-C software developers face the endless task of maintaining the views of their applications for GNUstep with Gorm and for Mac OS X with Interface Builder. As the application evolves and translations are put into place, this can become a major burden, slowing the application development.

Luckily, the Renaissance framework innovates by introducing many new concepts to help developers create portable applications efficiently. Among the innovations, are:

- Portability: the user interface can be reused entirely on platforms where Renaissance has been ported. At this time, it can be reused on top of GNUstep and Apple Cocoa.
- Localization: there is no need to duplicate the interface files under both GNUstep and Mac OS X. Renaissance integrates perfectly with Localizable.strings files.
- Intelligent autolayout mechanisms: each user interface component contains intelligence to automate alignments and autoresizing. These are essential especially when working with localizations.

- Unobtrusive integration: Renaissance integrates easily with current application code bases, as it uses the same paradigm of outlets and connections traditionally used by NIB files.

Unfortunately, Renaissance also has some drawbacks. For example, sharing a common user interface on both GNUstep and Mac OS X can lead to human interface guidelines (HIG) violations on both platforms. Moreover, it currently is not possible to use Cocoa-specific classes such as NSDrawer and NSToolbar with Renaissance.

In this article, we use the source code of the TIFF image viewer that was created for my previous article "Programming under GNUstep—An Introduction" [*LJ*, April 2003, /article/6418]. We replace the view, previously created with Gorm and Interface Builder, with one created in the Renaissance framework. You can download the source code of the application from the SSC FTP site [ftp.linuxjournal.com/pub/lj/listings/issue119/7102.tgz].

Installing Renaissance

In order to compile and install Renaissance under Linux, we first need to make sure GNUstep is installed properly. Using the latest stable release of GNUstep is highly recommended. At the time of this writing, these include GNUstep make 1.9.0, GNUstep base 1.9.0, GNUstep GUI 0.9.1 and GNUstep back 0.9.1. For detailed instructions on installing GNUstep, refer to the GNUstep Build Guide for UNIX Systems (see Resources).

Once GNUstep is compiled and installed, you must load the proper set of environment variables by executing a shell script. Bash users would use:

```
. /usr/GNUstep/System/Makefiles/GNUstep.sh
```

and C shell users would do:

```
. /usr/GNUstep/System/Makefiles/GNUstep.csh
```

Finally, to compile and install Renaissance, simply uncompress the Renaissance archive file and type (as root):

```
# cd Renaissance-0.8.0
# make
# make install
```

Under Apple Mac OS X, you either can install Renaissance from the source or use a precompiled version. To install it from the source, you first must install GNUstep make and then follow the same installation procedure as if you were installing it under GNUstep. Alternatively, you can download the binary version

from Renaissance's Web site, uncompress the file and move the resulting Renaissance.framework folder to your /Library/Frameworks/ folder. I personally recommend the latter option.

A Simple GNUstep Application

In the April 2003 GNUstep article, we developed a simple TIFF image viewer. For this application, we had to use Gorm under GNUstep and Interface Builder under OS X to build the user interface. Luckily, Renaissance's portability strengths can help us solve this burden. As our previous application uses the Model-View-Controller (MVC) design pattern, we easily can redo the view using Renaissance, as it already is well separated from the model and the controller.

The first step in redoing the view for our small application is to create the main gsmarkup file. A gsmarkup (short for GNUstep Renaissance Markup Language) file is a simple XML representation of how user interface elements should be created and how they should be placed on screen and connected with one another or other objects in the main application. For our application, the main gsmarkup file represents the view to be shared on both GNUstep and Mac OS X. To create it, open your favorite editor and create the TiffViewer.gsmarkup file containing the content shown in Listing 1.

Listing 1. TiffViewer.gsmarkup

```
<?xml version="1.0"?>
<!DOCTYPE gsmarkup>
<gsmarkup>
   <objects>
     <window id="window"
             title="Tiff Viewer"
             closable="yes">
       <hbox>
        <image id="imageView"
               scaling="toFit"
               hasFrame="yes"
               width="300" height="300"
               valign="expand" halign="expand"/>
       </hbox>
     </window>
  </objects>
  <connectors>
    <outlet source="#NSOwner"
            target="#window"
            key="window"/>
    <outlet source="#NSOwner"
            target="#imageView"
            key="imageView"/>
  </connectors>
</gsmarkup>
```

In Listing 1, we tell Renaissance to create a closable window with the ID window. Then, we place an image view inside this window with such initial properties as the width and height of 300 points. We also specify that we want this image view to be resizable horizontally and vertically. Renaissance

understands this and propagates the information up to the window to make it automatically resizable. We then define the connections for those two UI elements. We connect the window with our window outlet and the imageView with our imageView outlet. Those two outlets previously were defined in the file AppController.h.

Once the creation of the main gsmarkup file has been finished, we create the gsmarkup file (Listing 2) to hold the application menu used under GNUstep. In Listing 2, we define three menu items: Load Image, Hide and Quit. Each of them has an associated action that is invoked if the menu item is clicked on.

Listing 2. GNUstep-Menu.gsmarkup

```
<?xml version="1.0"?>
<!DOCTYPE gsmarkup>
<gsmarkup>
  <objects>
    <menu type="main">
      <menuItem title="Load Image"
                  action="loadImage:"/>
      <menuItem title="Hide"
                  key="h"
                  action="hide:"/>
      <menuItem title="Quit"
                  key="q"
                  action="terminate:"/>
    </menu>
  </objects>
</gsmarkup>
```

Once those two files have been created, we modify our initial GNUmakefile and replace the reference to the Gorm files with our two newly created gsmarkup files. We also add the Renaissance framework in our list of linked frameworks. The GNUmakefile now should look like Listing 3.

Listing 3. GNUmakefile

```
include $(GNUSTEP_MAKEFILES)/common.make

APP_NAME = TiffViewer

TiffViewer_OBJC_FILES = AppController.m ImageModel.m
TiffViewer_RESOURCE_FILES = TiffViewer.gsmarkup \
        GNUstep-Menu.gsmarkup
ADDITIONAL_GUI_LIBS += -lRenaissance
ADDITIONAL_OBJCFLAGS = -Wall -Wno-import

include $(GNUSTEP_MAKEFILES)/application.make
```

Then, we modify our initial TiffViewerInfo.plist to remove the reference to MainMenu.nib. The file now should contain the content shown in Listing 4.

Listing 4. TiffViewerInfo.plist

```
{
  ApplicationName = "Tiff Viewer";
  ApplicationDescription = "A small image viewer.";
}
```

The last step we must take before compiling the application is to implement two delegate methods in our application's controller. Those methods are responsible for loading the main gsmarkup file (TiffViewer.gsmarkup) and the one used for the application menu (GNUstep-Menu.gsmarkup). They need to be invoked automatically upon the application's startup on both GNUstep and Mac OS X. To do so, use an editor to open the AppController.m file and modify it so it has the content shown in Listing 5.

Listing 5. AppController.m

```
#import "AppController.h"

#import <Renaissance/Renaissance.h>

@implementation AppController
...
- (void) dealloc
{
  [model release];
  [super dealloc];
}

- (void) applicationDidFinishLaunching:
  (NSNotification *) theNotification
{
  [NSBundle loadGSMarkupNamed: @"TiffViewer"
            owner: self];
}

- (void) applicationWillFinishLaunching:
  (NSNotification *) theNotification
{
#ifdef GNUSTEP
  [NSBundle loadGSMarkupNamed: @"GNUstep-Menu"
            owner: self];
#else
  [NSBundle loadGSMarkupNamed: @"Cocoa-Menu"
            owner: self];
#endif
}

- (void) loadImage: (id)sender
{
  NSOpenPanel *oPanel;
  int result;

  oPanel = [NSOpenPanel openPanel];
...
```

Here, only two methods were added, -applicationDidFinishLaunching: and -applicationWillFinishLaunching:. This shows yet another strength of Renaissance—unobtrusive integration with current code bases.

Finally, compile and start the small application:

```
# make
# openapp TiffViewer.app
```

Once the application starts, click on the Load Image menu item and select a TIFF file. It should display the image properly in the window, as shown in Figure 1.

Figure 1. The Image Viewing Application on GNUstep for Linux

Apple Mac OS X Port

Under Mac OS X, we are sharing the main gsmarkup file with the GNUstep version of our application, so we now have to create the gsmarkup file used for our sample application menu, for Mac OS X. Doing so allows us to have a different menu for Mac OS X, which is required because the layout of menus under GNUstep (vertical) is different from the one on Mac OS X (horizontal). Create the file Cocoa-Menu.gsmarkup with the content shown in Listing 6.

Listing 6. Cocoa-Menu.gsmarkup

```
<gsmarkup>
  <objects>
    <menu type="main">
      <menu title="TiffViewer" type="apple">
        <menuItem title="Hide TiffViewer"
                  key="h"
                  action="hide:"/>
```

```
          <menuItem title="Quit TiffViewer"
                    key="q"
                    action="terminate:"/>
        </menu>
        <menu title="File">
          <menuItem title="Load Image"
                    action="loadImage:"/>
        </menu>
      </menu>
    </objects>
  </gsmarkup>
```

In Listing 6, we also define three menu items: Hide TiffViewer, Quit TiffViewer and Load Image. Contrary to GNUstep, we create the first two under the TiffViewer menu, to be displayed in bold (notice the `type="apple"`) and the latter under the File menu. We do this because the menu disposition on Mac OS X is different from GNUstep, and we want to follow the HIG at least with regard to the menus.

Once the file has been created, we need to create the Mac OS X project file and build the application. To do so, start the Project Builder application and proceed with the following steps:

1. From the File menu, choose the New Project... menu item and select Cocoa Application. Click on the Next button.
2. Specify the project name (TiffViewer) and the project directory, then click on the Finish button.
3. Select the Classes node in the Groups & Files panel and then click on the Add Files... menu item from the Project menu. Add the AppController.m and ImageModel.m files. Those are the same files used under GNUstep.
4. Expand the Other Sources node and delete the main.m file. We don't need this file.
5. Expand the Resources node and double-click on MainMenu.nib. This launches Interface Builder. From Interface Builder's MainMenu.nib window, delete MainMenu and Window by clicking on the corresponding icons and choosing Delete from the Edit menu. Save everything and then quit Interface Builder. We need to do so because Renaissance can provide the application menu using our gsmarkup file.
6. Select the Resources node and add the Cocoa-Menu.gsmarkup and TiffViewer.gsmarkup files, as you did in Step 3.
7. Expand the Frameworks and Linked Frameworks nodes and click on the Add Frameworks... menu item from the Project menu. Add the Renaissance.framework located in the /Library/Frameworks directory.
8. Finally, from the Build menu in Project Builder, choose Build and Run. This compiles and launches the application.

Figure 2. The Image Viewing Application on Mac OS X

As you have seen in this section, porting the application from GNUstep to Mac OS X is rather trivial. No code changes were required. As under GNUstep, you can load a TIFF file in the application and try to resize the window. You should see the image view automatically resizing both horizontally and vertically, as specified in the main gsmarkup file. You also should notice the Apple-style horizontal disposition of the application menu, as shown in Figure 2.

Translating the Application

As said before, Renaissance eases localization. In order to show how, let's translate our simple TIFF viewer to the French language. Renaissance automatically knows what to translate and what to maintain. In our menu gsmarkup files, each of the menu items had a title. Renaissance automatically uses the title of UI elements as a key in the Localizable.strings files to get the right translated string. In order to translate our sample application, create a French.lproj directory inside the project's root directory. In that newly created directory, create the Localizable.strings file with the content shown in Listing 7.

Listing 7. French.lproj/Localizable.strings

```
"File" = "Fichier";
"Hide" = "Cacher";
"Hide TiffViewer" = "Cacher TiffViewer";
"Load Image" = "Charger l'image";
"Quit" = "Quitter";
"Quit TiffViewer" = "Quitter TiffViewer";
```

We use the same file for both GNUstep and Mac OS X. Under GNUstep, modify the GNUmakefile to add instructions so that our translation resource gets installed. The following two lines need to be added to the GNUmakefile: `TiffViewer_LOCALIZED_RESOURCE_FILES = ...` and `TiffViewer_LANGUAGES = ....`

Listing 8. GNUmakefile Changes Required to Support Localization

```
...
TiffViewer_RESOURCE_FILES = TiffViewer.gsmarkup \
        GNUstep-Menu.gsmarkup

TiffViewer_LOCALIZED_RESOURCE_FILES = \
        Localizable.strings
TiffViewer_LANGUAGES = French

ADDITIONAL_GUI_LIBS += -lRenaissance
...
```

Finally, under GNUstep, recompile the application in order to copy the resource file properly and launch it using its French translation, like this:

```
# make
# openapp TiffViewer.app -NSLanguages '(French)'
```

On Mac OS X, you also have to create the French.lproj directory and the Localizable.strings file (or reuse the ones created for GNUstep) with the content shown in Listing 5. Once they have been created, follow these steps to activate the French localization in Project Builder:

1. Select the Resources node and from the Project menu, choose Add Files... and add the French.lproj/Localizable.strings file.
2. From the Build menu, click on Build.

To run the application in French under Mac OS X, from the System Preferences, click on the International icon. Then, drag Français before English and quit the application. From Project Builder's Debug menu, click on Run Executable. The application should start in French.

Conclusion

The Renaissance framework provides valuable innovations to help develop truly portable applications. Eventually, Renaissance will have a complete graphical editor, allowing you to create gsmarkup files graphically, as you can do now with Gorm, the Graphical Object Relationship Modeler for GNUstep, or with Interface Builder under Mac OS X.

In a future GNUstep article, we will enhance our simple TIFF viewer application to work with the GNUstep Database Library (GDL), an excellent free implementation of the NeXT's Enterprise Objects Framework (EOF).

Resources

GNUstep Build Guide for UNIX Systems: documents.made-it.com/GNUstep/Build

Objective-C Programming Language: developer.apple.com/techpubs/macosx/Cocoa/ObjectiveC

“Programming under GNUstep—An Introduction” by Ludovic Marcotte: /article/6418

Renaissance: www.gnustep.it/Renaissance/index.html

Source code of the application from the previous article: ftp.linuxjournal.com/pub/lj/listings/issue108/6418.tgz

Ludovic Marcotte (ludovic@inverse.ca) holds a Bachelor's degree in Computer Science from the University of Montréal. He is currently a software architect for Inverse, Inc., a small IT consulting company located in downtown Montréal.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The OASIS Standard for Office Documents: How All Users and Developers Can Benefit

Marco Fioretti

Issue #119, March 2004

A common set of file formats has the potential to be the most meaningful advancement for free software on the desktop.

Desktop integration begins with documents, not with any toolkit or bundle of applications. If files can be read and written by every application, users can communicate, work together and become integrated. In this sense, the OASIS XML format for office documents has the potential to be one of the most meaningful advances in free computing.

OASIS stands for Organization for the Advancement of Structured Information Standards. Formerly SGML Open, this nonprofit consortium, which includes such companies as IBM, Sun and Boeing, aims to create open standards for almost any kind of structured information. The one we cover here is an XML-based format common to all kinds of office files—text, spreadsheets, presentations and more.

The significance of an effort of this caliber to promote a file format, rather than any specific desktop, application or the Linux kernel itself, cannot be underestimated. Free as in free formats is even more important than free software. Only with them and the internal structuring that comes from XML can data be exchanged, with new or different programs without any need for converters, or be directly edited, indexed, analyzed and exchanged between heterogeneous groups or servers—like Web services without the hype. Data will start belonging exclusively to end users.

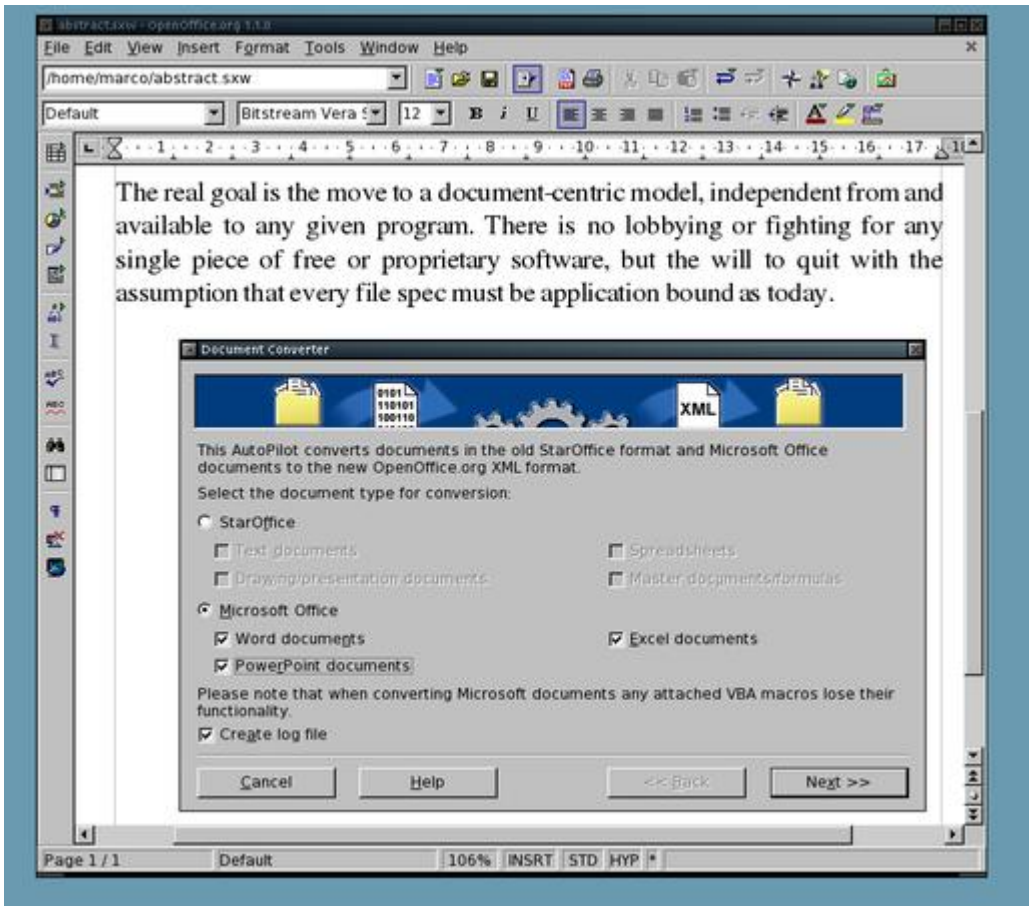


Figure 1. Switching to OASIS and never going back: OpenOffice.org can convert all your closed-format documents to the new standard with only a few clicks.

The OASIS Office Technical Committee had its first meeting in February 2003. The official file format should be voted on in February 2004. After the approval, Phase 2 will start; its main goal will be to extend the base specification to additional areas of application. The real goal is the move to a document-centric model, independent from and available to any given program, regardless of its license. The Technical Committee is determined to quit with the assumption that every file spec must be application-bound, as today.

Some farsighted public administrations already have started to think in this way. The Swedish Agency for Public Management says, “[We] should also follow and if possible support work that takes place in OASIS....An open file format for office software is of great importance for increased interoperability” (www.openoffice.org/servlets/ReadMsg?msgId=585772&listName=discuss). At the European Union level, IDA (Interchange of Data between Administrations) decided in 2003 to carry out exploratory work on open document formats and on how public administrations could persuade software vendors to support them.

What Does It Look Like?

The standard conforms to general W3C specifications for XML technologies and covers every aspect of document usages. User interaction, for example, is described in XML schema templates, which operate like traditional API functions. Even they, however, now are independent of any single application.

A text format can be much bigger and more inefficient than an equally free but binary one. Even when the performance hit would be noticeable, however, the benefits simply are too great to give up. In itself, an OASIS office file (be it text, presentation or spreadsheet) is a zip archive: the compression format chosen is a compromise of efficiency, speed of accessing internal parts and algorithm license. Unzipping it, we first find five XML files: styles.xml, presentation and formatting; contents.xml, actual contents; settings.xml, application settings such as zoom level and printer; meta.xml, language and uncoding metadata; and manifest.xml, an explanation of what all the other files are and their relative paths.

Other components (each in a predefined folder, so that even virus scanners have an easier time) may be macros, their dialogs and objects, such as charts or formulas.

Because the standard imposes that all pieces must be present in the zip archive, no information is lost: content, layout and everything else always travel together. Unlike some proprietary offerings in the same space, there is no restriction on which application must be employed to make full use of a document. WYSIWYG results are possible and can be specified fully or replaced in the styles.xml file. At the same time, however, content and presentation are decoupled; hence, content and nothing else is attainable by any application, for any conceivable use. kfile-plugin-ooo, for example, extracts all the metadata embedded in the new file format. The end user then can read, search by metadata or modify all this information straight from KOffice or Konqueror. This plugin also is included in the latest KOffice source trees.

Text format and internal structure make decades of UNIX experience in processing and generating text come back with a vengeance to tame complex, WYSIWYG office documents of every kind. Shell one-liners, Web spiders and so on can query and process directly, much like a database engine, single documents or whole classes of them. Viewing attached presentations as text in mutt or industry-level content management systems becomes easier. As a proof of concept, I was able to get the (admittedly rough) outline of Listing 1 from a presentation simply by typing:

```
# tr "<" "\012" < content.xml | grep ^text \
```

```
| cut '-d>' -f2, | uniq
```

Listing 1. Extracting the Text of an OASIS Presentation at the Command Line

```
Problems with
a lot of the bang up to date mainstream Free SW

Requires modern HW:

plenty of RAM

fast CPUs

big Hard disk drives

Unlike SW, modern HW cannot be "free as free beer"

Doesn't this sound familiar?
```

Encryption obviously is supported, and any paragraph can have an identity attribute. Through this feature, different users can be granted access to different parts of the same document based on their privileges. The default text encoding is UTF-8, even if other ones can be chosen. Suggestions to improve the standard can be posted to office-comments@lists.oasis-open.org.

What about End Users?

This design and implementations are all well and good, but users need some application to use it. What is available? Absolutely complete compatibility is possible only with software designed from scratch or with software that has been modified thoroughly to achieve it. In general, existing programs and their developers may have to compromise between the standard and their current concept of the perfect structure of the perfect document. For example, margins are a section or page property in some applications and a paragraph property in others.

This said, the users of OpenOffice.org will have the easiest time; the OASIS standard is built on and almost will be equal to the current OOo formats. AbiWord has both import and less-advanced export filters, but they are not 100% complete. Contributions to improve them are extremely welcome. This program also offers end users the option to use OOo as the default file format. The plan for KOffice, after improving the filters for version 1.3, is to start the switch to OASIS as the native format of future releases. David Faure, one of the chief KOffice developers, also is a member of the Technical Committee, and he foresees no real obstacles to a complete support of the standard, in spite of the frame-oriented rather than page-oriented paradigm used in KOffice.

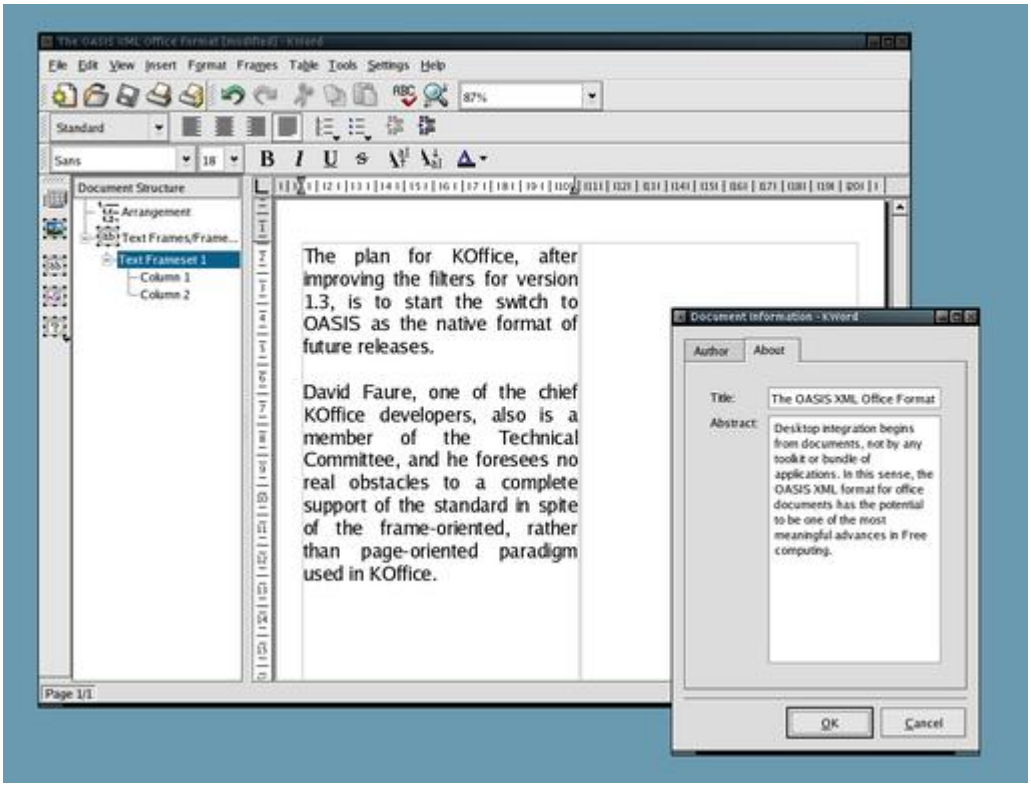


Figure 2. KOffice is ready to switch to OASIS fully. The internal structure already is stored in XML format, including metadata, and is searchable with external plugins.

SIAG offers some support for reading text and spreadsheet files through external applications, but nothing is available for writing. Emacs surely will come up with its own OASIS mode sooner or later, and WordPerfect also is officially represented in the Technical Committee. In short, things look good. Choice already is offered, and the only things left are to set OASIS as the default save format and to refuse to receive or send files in proprietary formats.

Developer Tools

A lot of code already is available to study and reuse for processing the OASIS file format. Whatever you choose, don't forget the standard itself and the main point—format and applications shall remain separated. If you want to improve the first, submit proposals as explained above. If you want faster or more featureful code, do it yourself or help the developer(s) of the corresponding application without touching the format or inventing a new one.

Several standalone filters already are available to move back and forth between OASIS/OOo files (or XML in general) and other formats. The utilities RTF2XML, ooo2txt, SIAG, O3read, o3totxt, o3tohtml, OOo2sDbk, Writer2LaTeX and soffice2html (see Resources) cover together RTF, (X)HTML, LaTeX, DocBook and, of course, plain text.

CPAN hosts several Perl modules useful for OASIS-related processing. OpenOffice::Parse::SXC parses OOo spreadsheets, making the text value of

each cell (but nothing else) available for the main script. It comes with a utility to convert OOo spreadsheets to CSV format. Another Perl module, XML::Excel can transform Excel spreadsheets into plain XML, dumping them into an intermediate structure for custom processing, if necessary. On the server side, Apache::AxKit::Provider::OpenOffice extracts the content of text (.sxw) files.

Tcl has linters, DOMs and XSLT interfaces, as well as an API that allows switching to different parsers with no changes to the application code. When nothing else is available, a native Tcl parser is used; otherwise the developer can take advantage of both Expat and Libxml (see below).

PDA developers have a dedicated project, related to OASIS quite directly, called XMerge that currently is developed in Java for Palm and Pocket PC. Its purpose is to allow the editing of OOo documents (maybe previously converted to a more limited format) with PDA native applications, in such a way that any changes can be merged back into the original format without loss of style, formatting and so on.

Parsers and Libraries

At a lower level, what is needed to manage OASIS files in a larger application, where the source language usually is C or C++ and the performance must be maximized? First of all, the program must include the proper library to compress and uncompress zipped files. This is not an OASIS-specific issue, so we won't deal with it further.

Once the single XML files are available, they have to be loaded in a way that understands and makes accessible the internal structure, that is, the relationships among the several elements. Once this step has been performed, data can be converted or processed in any manner. A lot of tools for this already exist. Several of them are designed to support general XML rather than OASIS, but the difference is quite a bit smaller than one might expect. And this situation is expected to improve soon after the standard is released.

Expat is a popular XML parser written in C that is basic and lacks a validation capability but still is the fastest one around. It also has front ends for practically every language. A more featureful library that supports DTD validation and is designed specifically for GNOME is Libxml. Like Expat, Libxml is written in C, is portable and can be used within a lot of languages. The Xerces parser, in Java, also can generate and validate XML documents.

In the Qt/KDE field, developers have at their disposal, besides the OOo plugin already mentioned, the related Qt classes and DOM implementation (QDom) to write or parse XML, as well as the KOffice DTD. At the time of this writing, these

tools still target the KOffice XML format, but they are expected to converge on the OASIS standard.

For security-conscious developers, the easiest starting point is the C XML security library (XMLsec), based on LibXML2, which supports both signing and encryption of XML material. SAXEcho is a (mostly) Java program that attaches itself to a running OpenOffice.org document to show the XML tree representation of the current document. It also validates or modifies the document operating directly on XML nodes, plus several other nifty things.

Event-Driven XML Processing

The parsers described above build an internal tree representation of the document. What should one do when developing applications that must deal with large documents? Keep in mind that large here means too big to fit into memory, which is not so big if this format must be usable even for low-end desktop applications.

The current solutions in this space follow the so-called SAX (simple API for XML) approach: instead of building the whole tree of a document in one fell swoop and keeping it there for further processing, go step by step. A SAX parser reads the document and, instead of keeping it all in memory, generates an event every time it finds something worthwhile. The parser then passes the event to event handlers that interact with the application. The something worthwhile can be XML document-type definitions, errors or elements of the actual content. A good starting point for SAX-based programming is the SAX Project. SAX2 already is supported in Java through JAXP and in Perl through the Orchard Project, which is quite stable, not to mention fast and lightweight, as far as SAX and XML processing are concerned.

Conclusion

All the research done for this article confirmed one of my first impressions: so far, the free software/open-source software approach to guarantee information interchange has been to develop cross-platform applications, which are difficult to maintain and optimize for each target environment. Now it looks like we are starting to do the right thing, which is to define truly Free, standard, toolkit-independent, cross-platform formats that leave everyone free to create any possible front end to read and write them.

Acknowledgements

Thanks above all to Gary Edwards and David Faure for all the material and explanations. Pierre Souchay (kfile-plugin-ooo) and the AbiWord developers also were very helpful.

Resources

AbiWord: www.abisource.com

CPAN: www.cpan.org

EU-IDA: europa.eu.int/ISPO/ida

Expat: expat.sourceforge.net

kfile-plugin-ooo: bad.sheep.free.fr/kfile-plugin-ooo.html

KOffice: koffice.kde.org

KOffice DTD: www.koffice.org/DTD/kword-1.2.dtd

Libxml: xmlsoft.org

OASIS Office File Format TC: www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

OASIS Web Site: www.oasis-open.org

OOo2sDbk: www.chez.com/ebellot/ooo2sdbk

ooo2txt: ooo2txt.free.fr

OpenOffice.org: www.openoffice.org

Orchard: orchard.sourceforge.net

QDom: doc.trolltech.com/3.1/xml-tools.html

RTF2XML: www.xmeta.com/omlette/rtf2xml

SAXEcho: xml.openoffice.org/saxecho

SAX Project: www.saxproject.org

SIAG, O3read, o3totxt, o3tohtml: siag.nu

soffice2html: hoopajoo.net/projects/soffice2html.html

Stop Word Attachments: www.gnu.org/philosophy/no-word-attachments.html

TclXML: tclxml.sourceforge.net

Writer2LaTeX: www.hj-gym.dk/~hj/writer2latex

Xerces: xml.apache.org/xerces-j

XMerge: xml.openoffice.org/xmerge

XMLsec: www.aleksey.com/xmlsec

Marco Fioretti is a hardware systems engineer interested in free software both as an EDA platform and, as the current leader of the RULE Project, as an efficient desktop. Marco lives with his family in Rome, Italy.

Archive Index Issue Table of Contents

Advanced search

Getting the Most from XMMS with Plugins

Dave Phillips

Issue #119, March 2004

Our readers' favorite audio tool offers some advanced features you can use to customize your listening experience.

XMMS (the X MultiMedia System) is a feature-rich multimedia player that has won the *Linux Journal*'s Readers' Choice Award for Favorite Audio Tool for the past four years. I assume, then, that the program needs little introduction here.

Most users of the program probably use it to play MP3s, but its default range of supported file formats also includes OGG, WAV, CD audio and all formats supported by the MikMod music module player. Thanks to various third-party input plugins, XMMS also can play almost every available audio and video format, as we soon shall see. Supported audio output formats include ALSA, OSS, esd and aRts. It also has a driver for writing audio output to disk in the WAV sound file format, which is quite handy for converting sound files for burning to an audio CD.

But, XMMS is more than a versatile player. This brief article dives a little deeper into the program to expose some of its extended features, such as its equalizer and playlist capabilities, as well as some interesting and useful third-party plugins.

Basics

XMMS is divided into three panels: the player, a graphic equalizer and a playlist window. The player contains the expected transport controls as well as sliders for volume and balance, or pan position. Another larger slider controls a pointer for random access into the file being played. Toggles are provided for the other panels, and switches control song looping and random playback.

The graphic equalizer (EQ) presents 11 sliders, one for preamp gain (+/– 20db), the others for boosting or cutting audio frequencies in ten bands (channels).

Ten channels provide decent basic equalization, with starting frequencies at 60, 170, 310, 600, 1k, 3k, 6k, 12k, 14k and 16k Hertz. These numbers indicate the throw (movement range), for each slider ranges from 100 up to 6,000 Hertz, which is rather coarse-grained, but the sliders are nicely responsive with smooth real-time audio updating. The default equalizer is fine for most desktop audio listening, but for finer resolution you can try Felipe Rivera's graphic EQ plugin that can be configured for 10, 15, 25 or 31 bands (see this article's Resources for links to all software discussed).

The playlist panel displays the files to be played and provides a number of useful file controls. The buttons on the lower left add and delete files from the list, make and sort selections and open an ID3 editor for the information tag attached to an MP3. The playlist window also includes a set of transport controls identical to those in the player panel, along with indicators for elapsed time during file play and for total time of the playlist. Finally, the List button on the lower right lets you load a new playlist, blank out the current list or save the contents of the playlist to a new file. Playlists are plain-text files in the common M3U playlist format, meaning the entries are simply paths to the included files. The playlist is quite flexible and accommodates any file type supported by XMMS and its plugins.

A Little More Advanced

Let's take a peek at some of these plugins. XMMS utilizes plugins for file I/O, special effects, visualization and a general category. For the purpose of this article, I focus on only a few I/O and effects plugins.

Erik de Castro Lopo's libsndfile has taken its place as the preferred sound file I/O library for such projects as the Ardour digital audio workstation and the MusE audio/MIDI sequencer. Erik's libxmms_sndfile plugin expands XMMS' sound file support to a wider variety of sound file types, including AIFF, AU/SND, IRCAM SF and many others. Nandan Dixit's libxmmsmplayer plugin adds the ability to play any video file format supported by the MPlayer video playback engine, extending XMMS' file support to MPEG, ASF, AVI, MOV and other video formats. Nick Lamb's ladspa.so is in fact a plugin to host plugins. LADSPA (the Linux Audio Developers Simple Plugin API) is an interface designed for creating simple but powerful audio processing plugins. Ladspa.so brings dozens of interesting effects to XMMS, all usable during real-time playback. Figure 1 shows off XMMS with a playlist containing sound files in five formats (AIFF, AU, MP3, OGG and WAV); movies in MPEG, AVI and RM formats (yes, the MPlayer plugin handles RealVideo too); and the LADSPA plugin at work lending its plate reverb and retro flanger to the XMMS audio processing functions.

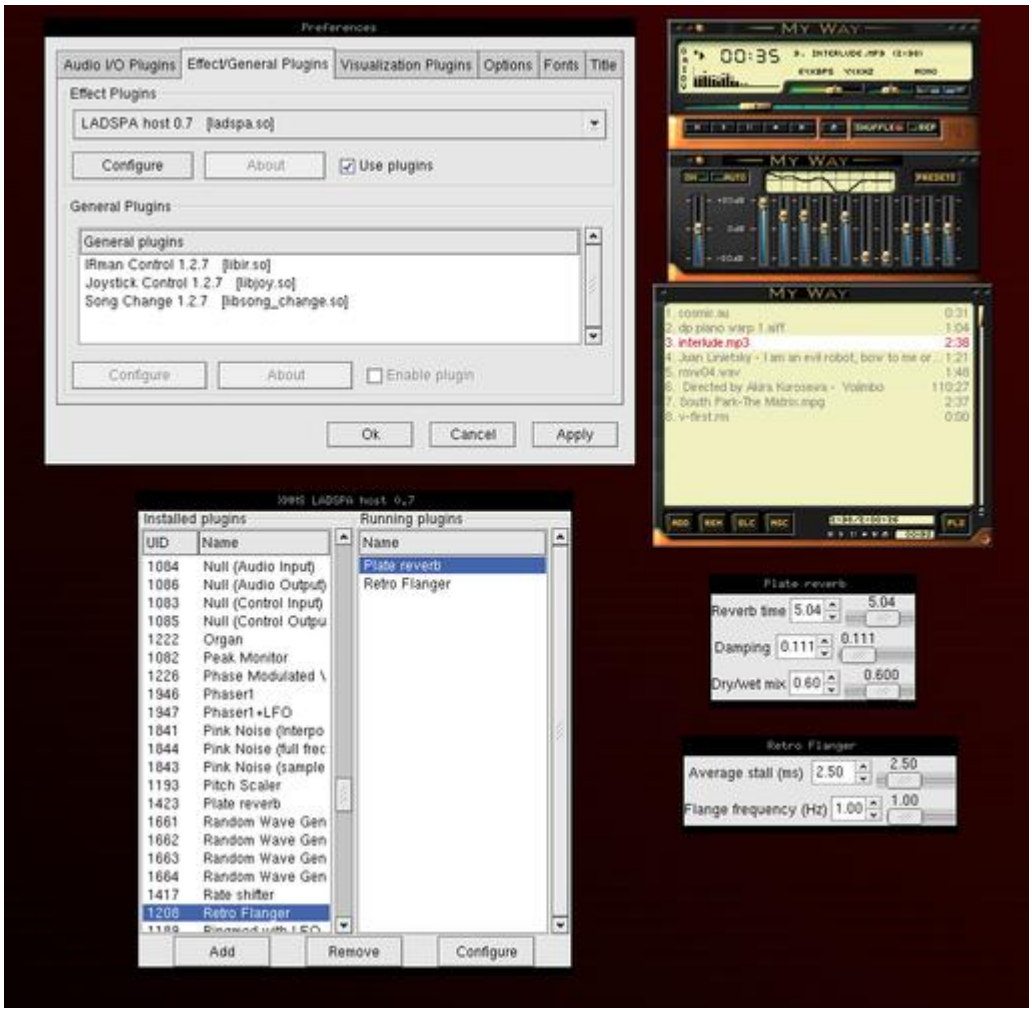


Figure 1. XMMS Showing a Playlist Containing Sound Files in Five Formats

Finale

That's all for this preview of some of XMMS' advanced features. Big thanks to the main development team for creating and maintaining XMMS, to the whole crew of third-party plugin developers for their wonderful expansions and to 4Front Technologies for its support of this most excellent Linux multimedia software.

Resources

Felipe Rivera's Equalizer Plugin: equ.sourceforge.net

The LADSPA Plugin: www.ecs.soton.ac.uk/~njl98r/code/ladspa

The libsndfile Plugin: www.zipworld.com.au/~erikd/XMMS

The MPlayer Plugin: xmmsmplayer.sourceforge.net

XMMS: www.xmms.org

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 in *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Manipulating OOo Documents with Ruby

James Britt

Issue #119, March 2004

Who says you have to wait for some future OS to integrate your office documents with business applications you develop? Work with OpenOffice.org's XML-based documents using Ruby.

OpenOffice.org (OOo), a featureful suite of office tools that includes applications for word processing, spreadsheet creation and presentation authoring, has seen an increase in enhancements and overall quality. OOo lives up to its name by making both source code and file formats completely open. This is a big plus for anyone wishing to manipulate documents without needing to have the creator application present.

In general, two ways exist to access or manipulate document content. One is to automate the source application, letting a program substitute for a person entering commands. The other is to go directly to the document. An advantage of the first approach is you get to exploit the power of an existing application, saving yourself a good deal of time figuring out file formats and processing commands. OOo can execute internal macros and expose a scripting interface through UNO. The downside is you need to have the actual application handy, and even then it may not be able to do what you want. This article describes the second approach: accessing and manipulating documents by going directly to the source.

OOo Extract

I first became aware of what could be done with an OpenOffice.org document when Daniel Carrera announced his OOoExtract program. This is a Ruby application that allows you to run command-line searches of OOo Writer document content. As the home page states, OOoExtract performs matches on both text content and styles, executes search patterns using full regular expressions and runs searches built with Boolean operators. The program runs

on any platform that has a Ruby interpreter, and they are available for pretty much every OS around.

Ruby has been discussed before in *Linux Journal*, but if you are not familiar with it, a good though brief description might be to say it's a cross between Perl and Smalltalk, with some features from Lisp and Python. It is deeply object-oriented and has a clean intuitive syntax. Yukihiro “Matz” Matsumoto, its creator, released the first alpha version in 1994. It has grown steadily in popularity, and the Third International Ruby Conference was held in November 2003, in Austin, Texas.

To get a feel for OOoExtract, download the program; currently, you can get the application as a single executable file or as a tarball with constituent libraries in separate files. Once installed, we can create a simple Writer document and run some searches. If you have OOo handy, fire it up and enter some brief text, such as:

```
My sample document
It has two lines
```

Save the file as sample1.sxw to the same directory where you installed OOoExtract, and run OOoExtract from the command line, like this:

```
./ooo_extract.rb --text sample sample1.sxw
My sample document
```

The program searches sample1.sxw for any lines that match on the word sample. Actually, this is a regular expression, albeit a simple one. We also can use more complex expressions, such as this one that matches any three-letter word:

```
./ooo_extract.rb --text "\s\w\w\w\s"  sample1.sxw
It has two lines
```

This is all well and good, but OOoExtract really shines by letting us search on content metadata, the extra information about the text in our document. Suppose we add an additional line to our sample Writer document:

```
This one has some extra formatting
```

After entering the text, select the word extra and apply the Footer paragraph style. Save the file and run this search:

```
./ooo_extract.rb --style="Footer"  sample1.sxw
This one has some extra formatting
```

In addition to locating text based on content, OOoExtract also can give you text with specific markup. This is quite handy if you create your own semantically rich styles. You then can use OOoExtract to retrieve information based on content and meaning, effectively turning an OpenOffice.org Writer document into a lightweight database. You can run the program against multiple files by using wild cards in the filename. For example, suppose you store recipes in Writer files. If you've defined and used custom styles, you could locate specific information, such as what recipes have apples as an ingredient:

```
./ooo_extract.rb --text="apple" --style="Ingredient" recipes/*.sxw
AppleSalsa.sxw: 2 medium red apples
AppleStrudel.sxw: 4 cups peeled and sliced apples
```

The SXW File Format

So, how does OOoExtract do its magic? The secret is in the file format. Although any given Writer file has an sxw file extension, running the UNIX file command tells us that it is a zip file:

```
$ file sample1.sxw
sample1.sxw: Zip archive data, at least v2.0 to extract
```

And what has been zipped? Let's see:

```
$ unzip -l sample1.sxw
Archive:  sample1.sxw
  Length     Date   Time    Name
 --------    ----   ----    ----
       30  11-26-03 01:40   mimetype
     2328  11-26-03 01:40   content.xml
     8358  11-26-03 01:40   styles.xml
     1159  11-26-03 01:40   meta.xml
     7021  11-26-03 01:40   settings.xml
      752  11-26-03 01:40   META-INF/manifest.xml
 --------                   -------
    19648                   6 files
```

The OOo XML format exposes all content and metadata in plain text; there is no need to worry about cryptic binary encoding or complex layout. Because the data is exposed as XML, numerous existing XML tools are available for extra OOo parsing. Having the file in plain text means, of course, that anything you might want to know about the file is available if you simply look. However, we get a good deal of help because the OpenOffice.org team also provides assorted documentation detailing the format. The technical reference manual for OpenOffice.org XML File Format 1.0 is a 571-page PDF document. I confess to not having read the entire tome, though I doubt it lacks any detail one might care to find.

For our purposes, we need look only at some basic markup to see how OOoExtract works and to gain some understanding of the markup.

If you unzip our sample document and load content.xml into a text editor, you should notice a few things. First, the file is not formatted for your viewing pleasure. You may want to run the file though an XML-formatting tool, such as tidy, to get some new lines and indentations in place to make it easier to follow.

The file starts with an XML declaration, followed by a DOCTYPE reference. Right after that comes the root element, office:document-content. The beginning tag has a good number of XML namespace attributes. We needn't be concerned with these, but they give some idea of the range of content one might find in an OOo document.

Immediately inside the root element we find child elements for scripts, font declarations and styles. As ours is a fairly simple document, the data here is sparse. For our immediate interests, the useful stuff comes inside the office:body element. Yet, even here, a few elements simply declare the presence (or, in our case, the absence) of various items, such as tables and illustrations. The full document is available from the *Linux Journal* FTP site [ftp.linuxjournal.com/pub/lj/listings/issue119/7236.tgz].

The real content in our document appears inside of text:p elements:

```
<text:p text:style-name="Standard">My sample
document</text:p>
<text:p text:style-name="Standard">It has two
lines</text:p>
<text:p text:style-name="Footer">This one has
some extra formatting</text:p>
```

Incidentally, if you are unfamiliar with some of the details of XML syntax, this notation simply says that it is a p element, defined in the text namespace. The use of the prefix and colon is a shorthand way to reference the namespace URI given at the top of the document. It's used to avoid name collisions with other p elements that may be defined for some other XML vocabulary. For our purposes we can simply think of it as one complete element name.

Our sample document had only three paragraphs, so as we might expect, there are three text:p elements. Each one has a text:style-name attribute that indicates a style to apply to the text. It is this attribute that lets OOoExtract locate text based on styles.

You may be wondering about the Footer style. Our content.xml file does not define it, and indeed this separation of style name from implementation detail is good. It would be a shame if instead of a simple name, the document had assorted attributes for font size and family, color and so on. The ability to locate content based on semantic or structural data would be lost, and we would be confined to treating the data strictly in rendering terms. If you really do want to

see how OOo defined the Footer style, you can peer into styles.xml. There you'll find that Footer is based on the Standard style, with a few changes.

From Zip to REXML

It's all well and good that OpenOffice.org uses zipped XML, but once we've extracted these files, what is next? Lucky for us, Ruby 1.8 includes an outstanding XML parser, REXML. REXML is an XML 1.0 conformant parser, and in addition to its own Ruby-style API, it provides full implementations of XPath and SAX2. It was developed and is maintained by Sean Chittenden. Sean says he wrote REXML because, at the time, there were only two choices for XML parsing with Ruby. One was a binding to a native C parser, a possible limitation on portability. The other was pure Ruby, but in Sean's view, it lacked a suitable API. Sean was familiar with various Java XML parsers but disliked their adherence to the W3C's DOM or the community-driven SAX. The designers of Electric XML offered an API based on known Java idioms, one that readily would be intuitive to Java programmers.

Such was the philosophy behind the REXML API; the name stands for Ruby Electric XML. Not surprisingly, though, the REXML API moved from the Java-flavored original to a Ruby-way design, allowing developers to access and manipulate XML using the syntax and features, such as blocks and built-in iterators, common to Ruby.

The REXML API

The REXML tree parser easily lets one load XML documents:

```
require "rexml/document"
file = File.new( "som_xml_file.xml" )
doc = REXML::Document.new file
```

or:

```
require "rexml/document"
my_xml_string = "<sample>
   <text>This is my REXML doc</text>
   </sample>"
doc = REXML::Document.new my_xml_string
```

The Document constructor takes either a string or an I/O object; REXML figures out which it is and does the right thing. Once you have a document, you can locate elements using Ruby's Array and each syntax combined with an XPath selector:

```
my_xpath = "sample/text"
doc.elements.each( my_xpath ){
    |el| puts el.text }
```

In the above example, the each method iterates over each element matched by the XPath selector. A code block (the part inside the { ... }) is called for each iteration. The variable el is the current element in the iteration, so this example simply prints the text for each element matched by the XPath.

XPath

Our sample Writer document and its corresponding XML is quite simple, so finding what we want is close to trivial. It wouldn't take much to figure out the right element for particular content. A simple example can be best for articles such as this, but in real life we aren't likely to see anything that basic. We may know only limited details of the markup, such as the style attributes or a parent element. Finding such content becomes more of a challenge, but XPath helps save the day.

XPath is a W3C recommendation for addressing parts of an XML document. It allows one to construct a path specifier that defines location based on element and attribute names and content, plus relative or absolute positioning. Given a complex XML document, you can define an XPath expression that locates, for example, all text:p elements that are immediate children of the office:body element with this expression:

```
*/office:body/text:p
```

The leading asterisk says (in XPath-speak) to follow any path through the XML document tree that leads to a text:p element that is the child of an office:body element. With REXML, we can use this XPath to retrieve and iterate over a collection of matching elements:

```
xml.each_element( */office:body/text:p" ) do |el|
   # do something with el, such as
   # look for content or a style attribute
end
```

In this example, the code between do and end is a block. It is like an anonymous function that gets called for each item in the collection—in this case, each element matching the XPath—where the item is passed in as an argument, indicated by the two vertical bars just after “do”. This is essentially how OOoExtract works, but you should visit the OOoExtract home page for details on the numerous command-line parameters.

Toward a More General OOo API

Having seen OOoExtract, I wanted to have a more general-purpose OOo object for Ruby. The same basic ideas that drive OOoExtract could allow not only reading data, but creating, updating and deleting, for example, the CRUD operations we know and love from database tools. To this end, a project named OOo4R has been created on RubyForge, the Ruby software CVS repository. The design goals are simple access to data and metadata, transparent use of XPath and an intuitive API for doing the commonplace, such as adding paragraphs, headings and styles. Space does not allow a complete walk-though of all such features, but we can look at accessing document metadata to see one way of using Ruby's dynamic message handling to extract element content.

Earlier we saw that an OOo document has several XML files packaged in a single zip file. We looked at the content.xml file; another is meta.xml. It holds information about the document itself, such as the document title, the creation date and the word count. The root element is office:document-meta. This, in turn, contains an office:meta element that holds numerous child elements with the data of interest. For example:

```
<meta:initial-creator>James Britt
</meta:initial-creator>
<meta:creation-date>2003-11-25T17:36:31
</meta:creation-date>
<dc:creator>James Britt</dc:creator>
<dc:date>2003-11-25T18:40:59</dc:date>
<dc:language>en-US</dc:language>
<meta:editing-cycles>13</meta:editing-cycles>
```

The full metadata file is available from the *Linux Journal* FTP site [ftp.linuxjournal.com/pub/lj/listings/issue119/7236.tgz].

In addition to a main Document class, OOo4R defines a meta class to encapsulate the metadata. A meta class uses an REXML document to hold the contents of meta.xml. A meta object largely is a collection of attributes. Typical usage either would be asking an object for a particular value, such as the name of the author, or assigning a value, such as a new title. One way to code this would be to write a series of explicit attribute accessor methods. We would need two methods for every attribute. Or, we could use dynamic method invocation by grabbing accessor messages, finding a matching meta attribute and either performing the requested action on the corresponding attribute or raising an exception.

The following code example focuses on the Dublin Core metadata elements used in OOo. The Dublin Core Metadata Initiative is an open forum for defining metadata standards. Dublin Core elements often can be found in RSS feeds

and some XHTML documents. As with all elements in an OpenOffice.org XML file, the elements have a namespace prefix. Rather than have users know and use these prefixes, we can map the full element name to something friendly.

The definition of the Meta class begins with the creation of a hash that maps friendly names to actual element names, plus a class constant to hold the base XPath for the metadata. The class constructor simply creates an REXML document from the XML source:

```
module OOo
  class Meta

  NAME_MAP = {
   'description' => 'dc:description',
   'subject'     => 'dc:subject',
   'creator'     => 'dc:creator',
   'author '     => 'dc:creator',
   'date'        => 'dc:date',
   'language'    => 'dc:language',
   'title'       => 'dc:title'
  }
    XPATH_BASE  = "*/office:meta"

    def initialize( src )
      @doc = REXML::Document.new(  src.to_s )
    end
```

We can redefine the method_missing method available to all Ruby classes so that, rather than raising an exception (as it would do by default), it looks to see if the message sent to the object maps to some item in our metadata:

```
def method_missing( name, *args )
  n = name.to_s
  if is_assignment? n
    el = map_for_assignment n
    xpath = "#{XPATH_BASE}/#{el}"
    assign( xpath, *args)
  else
    el = Meta.map_name n
    xpath = "#{XPATH_BASE}/#{el}"
    find( xpath  )
  end
end
```

The first argument to method_missing is a symbol object, so our code grabs the string representation. The is_assignment method simply checks if the name ends with an = character. If this is an assignment request, then map_for_assignment removes any trailing characters following the metadata name and maps the friendly name to the actual Dublin Core element name; assign updates the corresponding element in the REXML document:

```
def assign( xpath, val )
  node = @doc.elements.to_a( xpath )[0]
  node.text = val
end
```

If this does not appear to be an assignment, the code tries to read some metadata. As before, the name is mapped, but now the code calls find:

```
  def find( xpath )
   begin
    return @doc.elements.to_a( xpath.to_s )[0].text
   rescue Exception
    raise OOo::OOoException.new(
       "Error with xpath '#{xpath}': #{$!}", $@ )
   end
  end

  # Helper methods omitted ...

   end
  end
```

The technique works for accessing the other metadata elements, though there are special cases where the metadata is contained in a series of child elements. Updating the zip file contents and writing the zip file back to disk using Ruby's built-in Zip class, lets us save modified OOo documents.

Summary

Because the OpenOffice.org file format uses a fully documented XML format, OOo files may be created or manipulated without requiring OOo itself. Ruby's built-in XML handling and dynamic nature make it a natural fit for OOo tasks.

Resources

Dublin Core: dublincore.org

OOo_extract: www.math.umd.edu/~dcarrera/openoffice/tools/ooo_extract.html

OOo Formats: xml.coverpages.org/starOfficeXML.html

OpenOffice.org XML: xml.openoffice.org

Ruby: linux.oreillynet.com/pub/a/linux/2001/11/29/ruby.html

RubyForge: www.rubyforge.org

“Thinking XML: The open office file format”: www-106.ibm.com/developerworks/xml/library/x-think15

XPath: www.w3.org/TR/xpath

James Britt runs Neurogami, LCC, a software and design company in Scottsdale, Arizona. He has coauthored a book on XML for the Wrox Press, written various articles on software development and gave a presentation on Ruby and XML at

the Third International Ruby Conference in Austin, Texas. He can be reached at jamesgb@neurogami.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

GUI Scripting with Tcl/Tk

Derek Fountain

Issue #119, March 2004

Tcl/Tk may not look very modern, but it has handy features such as variables that automatically take on the value of a widget. Tcl/Tk remains the tool of choice for many script writers.

Although many Linux developers are only now discovering the combination of a scripting language and a graphical user interface (GUI) toolkit, this sort of development environment is not new. The largely unsung forerunner to projects like PyQt and pyGTK is Tcl/Tk, the first footprints of which can be traced back to before Linux even was created. Supported by an enthusiastic community, Tcl/Tk quietly and efficiently has been providing cross-platform GUI scripting to UNIX, Windows and Macintosh developers for many years.

The language itself currently is up to version 8.4.5.0, and the Tcl/Tk application development tool of choice, Visual Tcl, recently has been updated to version 1.6 after two years of development. This article looks at the language, toolkit and Visual Tcl and describes how they can be used to produce a neat solution to a real requirement.

An Overview of Tcl/Tk

Although somewhat trampled in the stampede script writers made toward Perl when a scripting language was required to drive the emerging Internet, Tcl still is a technical match for Perl, Python or any other comparable language. Often described as the best kept secret of the Internet, it is a free (in all the best senses of the word), full-featured language driven by a byte code compiler that produces performance on a par with any of its peers. It is used in all the places other scripting languages are used: system administration, task automation, server back ends and, as we shall see shortly, application development.

As a programming language, Tcl is exceptionally easy to learn. In contrast to the complicated feature sets and syntaxes of Python and Perl, Tcl is procedural and

straightforward in nature. The entire syntax is described in exactly 11 rules, from which the whole language is built. Ironically, it's this simplicity that sometimes confuses people who are new to Tcl. An experienced programmer can learn to read Tcl scripts in ten minutes and write them inside an hour. A beginner doesn't take much longer.

Documentation is top rate, coming in the form of comprehensive and well written man pages. A complete HTML package of the documentation also is available. If man pages are a little intimidating for the new user, a decent selection of books exist for Tcl/Tk, the pick of which probably is Brent Welch's recently updated *Practical Programming in Tcl and Tk* from Prentice-Hall PTR. Also worth a mention is the Tcler's Wiki, which is one of the largest and best supported wikis anywhere on the Internet.

Tcl philosophy centers on one idea: it's an extendable language. Most languages allow a developer to write functions and procedures, but Tcl goes much further. Tcl allows developers to extend the entire language with new commands and functionality, up to and including adding fundamental language structures such as object orientation. The Tk toolkit actually is another optional extension to the Tcl language, which happens to provide a whole set of Tcl commands to create, drive and control GUI widgets. Like dozens of other extensions, Tk has long been included in the Tcl core distribution and now is seen more as a part of the language than an extension of it.

The Project

In order to test-drive the latest versions of Tcl/Tk and Visual Tcl, I needed a small project to develop. A personal requirement provided just the thing. Since getting a digital camera, I've often wanted to throw a couple of pictures onto a Web page quickly so that friends and family could see them. A full-blown Web gallery application would be overkill; I need only the ability to select one or two image files, add a few lines of text and then have a single Web page appear that I can upload to a Web server. Figure 1 shows an example of the sort of page I would like to be able to produce.

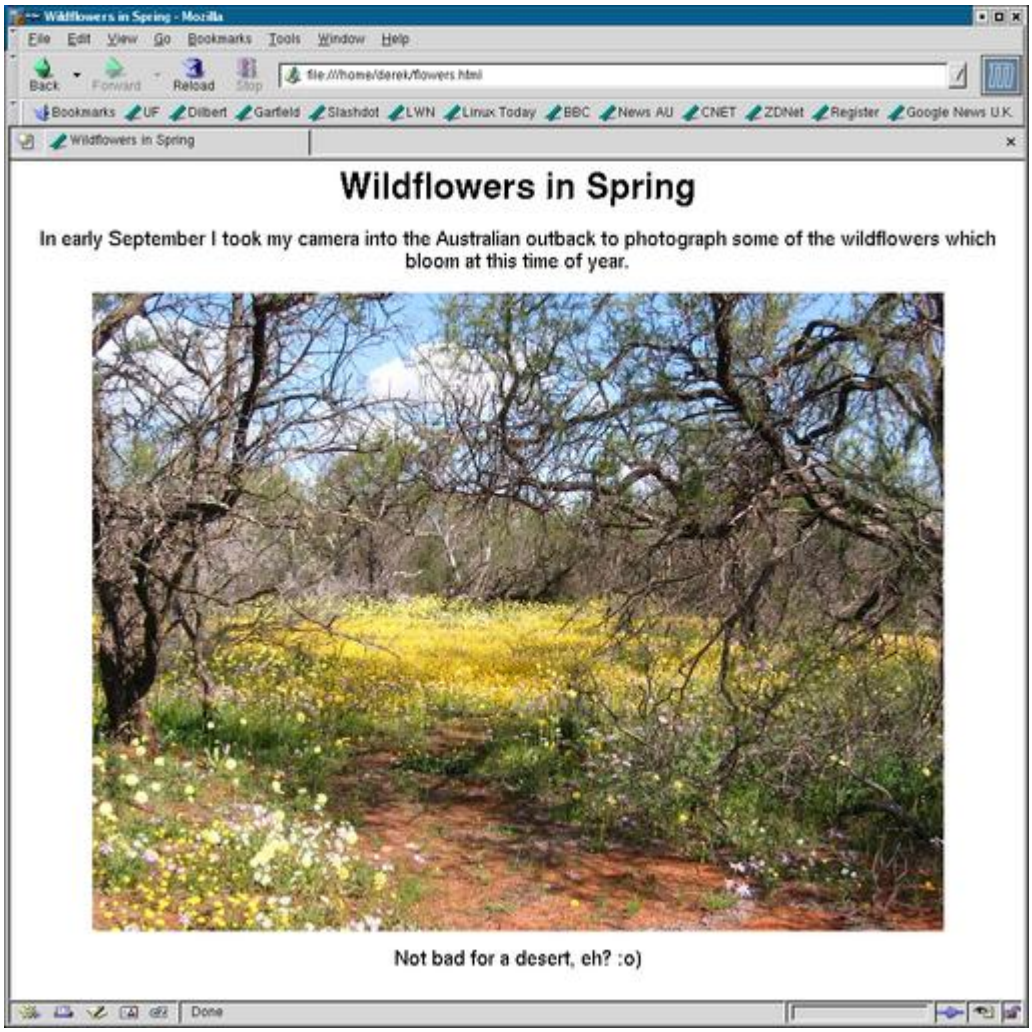


Figure 1. The task is to produce simple Web pages containing images and a small amount of text quickly, like this one.

This sort of project is an ideal candidate for a GUI-based script. It's a fairly simple task that isn't dependent on speed but that clearly benefits from having a graphical user interface. The function of the GUI is simple: present users with an interface where they select some image files, viewing them if necessary, and collect a few lines of accompanying text. The script then can use a standard tool to produce the HTML page. In this case, that tool is the XSLT processor from the libxml2 package found on almost every modern Linux system.

The rest of this article looks at how the combination of Tcl/Tk and Visual Tcl were used to develop this little application rapidly. Figure 2 shows the final script in action; the code can be downloaded from the link provided at the end of this article.

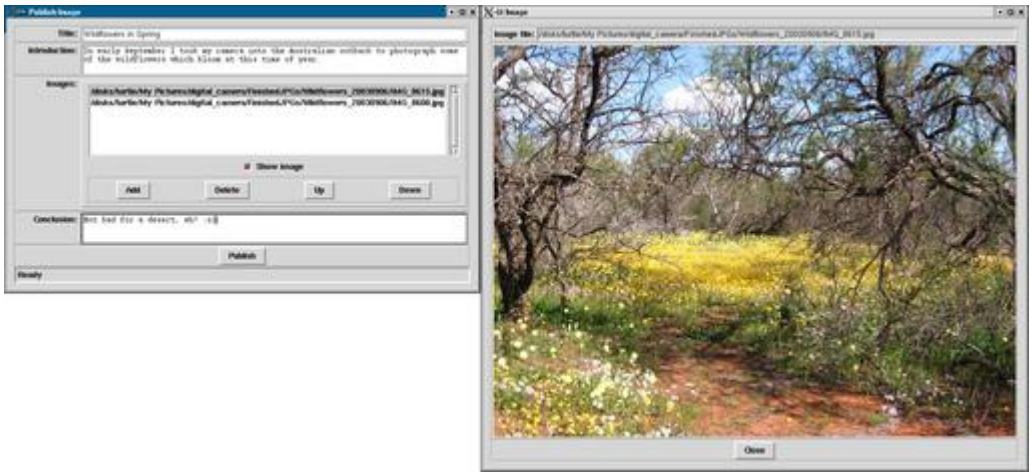


Figure 2. The script is running, with the image display window open.

Getting the Software

Most Linux distributions come with Tcl/Tk. However, I always install and use the latest version of ActiveTcl from ActiveState, Inc. Apart from being up to date and professionally presented, it provides a standard Tcl package with a lot of useful extensions. If you know your users are using ActiveTcl, you know exactly which extensions they have on their machine and therefore can guarantee your script can run. I encourage anyone who wants to run the project in this article to download and install ActiveTcl-8.4.5.0 or later, as that's what I used for development. ActiveTcl comes with its own installer, and if you install it in, for example, /opt/ActiveTcl-8.4.5.0, it doesn't interfere with any existing Tcl/Tk installation. If you already have a Tcl/Tk package in /usr/bin, ensure you set an early entry in your user account's PATH to point to the ActiveTcl bin directory.

Visual Tcl is available from SourceForge and also comes with its own installer. Many Linux distributions include it, but make sure you have the latest version.

Developing a Tcl/Tk Script

A common approach to Tcl/Tk scripting is to start by designing the GUI. This process allows the developer to think through all the features the application requires, and it produces a solid framework on which those features can be built. When things start getting complicated, this approach breaks down and something more formal, like a Model, View, Controller pattern, is required. But for small applications, such as my Web page, or for rapid prototyping, getting a GUI together is a good starting point. So I'll start with Visual Tcl.

A Look at Visual Tcl

The days when developers would sit at a text editor manually arranging buttons, listboxes and other widgets by brain power alone are pretty much gone. This is the sort of job that should be done with a graphical tool. Dragging

and dropping widgets makes development much quicker, especially for beginners.

Visual Tcl provides exactly these sorts of facilities and then some. In fact, it doesn't seem too sure whether to behave like a cut-down integrated development environment (IDE). It occasionally offers a text editing window where the user can write the Tcl code that forms the actual application, rather than limiting itself to dealing with the development of the GUI. On the other hand, it doesn't offer a debugger or other traditional IDE features, so it's difficult to justify calling it a real IDE. I dealt with this confusion of personality by going into the configuration dialog for the application and switching off many of the features that seem to get in my way (Figure 3).

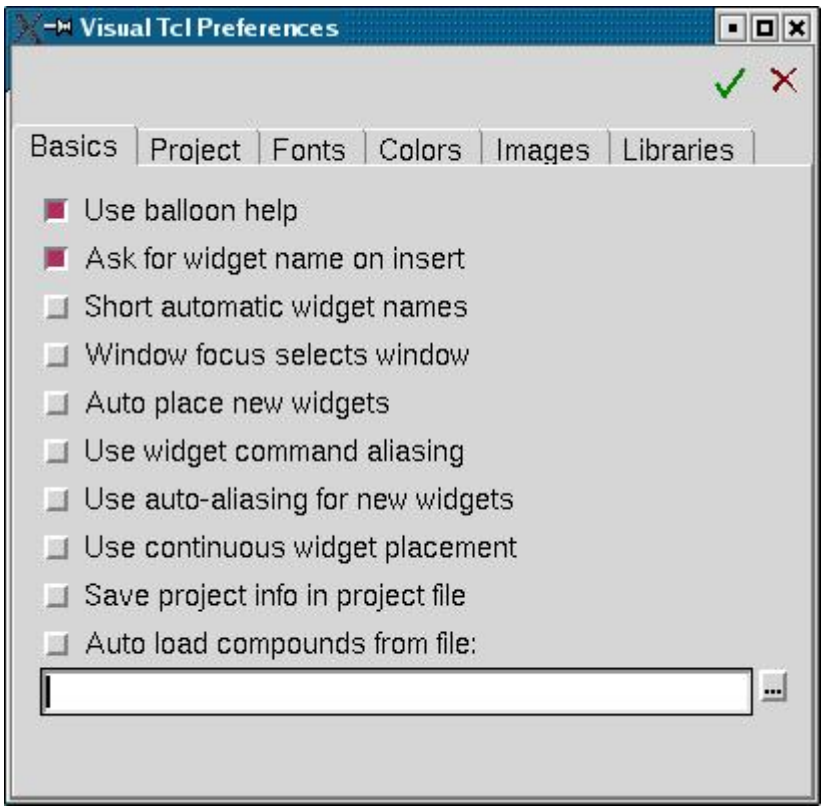


Figure 3. Visual Tcl is highly configurable.

Instead I chose to write the bulk of the application logic in my favoured environment (XEmacs) and simply used the output from Visual Tcl as a library that creates the GUI for my script. Credit goes to Visual Tcl for being flexible enough to be used in the way of my choosing. Listing 1 shows my wrapper script, which is the starting point for the application code itself.

Listing 1. A simple wrapper to keep the Visual Tcl code (in gui.tcl) separate from the main script. The #! line weirdness is a common way of starting a Tcl/Tk script.

```
#!/bin/sh
```

```
# the next line restarts using wish \
exec wish "$0" "$@"

#
# My own procedures and "pre-gui" code will go here
#

# Load and run the GUI code created by Visual Tcl
#
source gui.tcl

#
# Any "post-gui" code I need can go here
#
```

Once I understood the way I wanted to work with the tool, it didn't take too long to produce the output I wanted. Widgets are placed using a simple point-and-click interface, and a separate Attribute Editor window allows for the fine detail of widget behavior to be tweaked and fiddled with to the heart's content. Tk widget layout devices also are easy to control when you understand them. Figure 4 shows the Visual Tcl development environment.

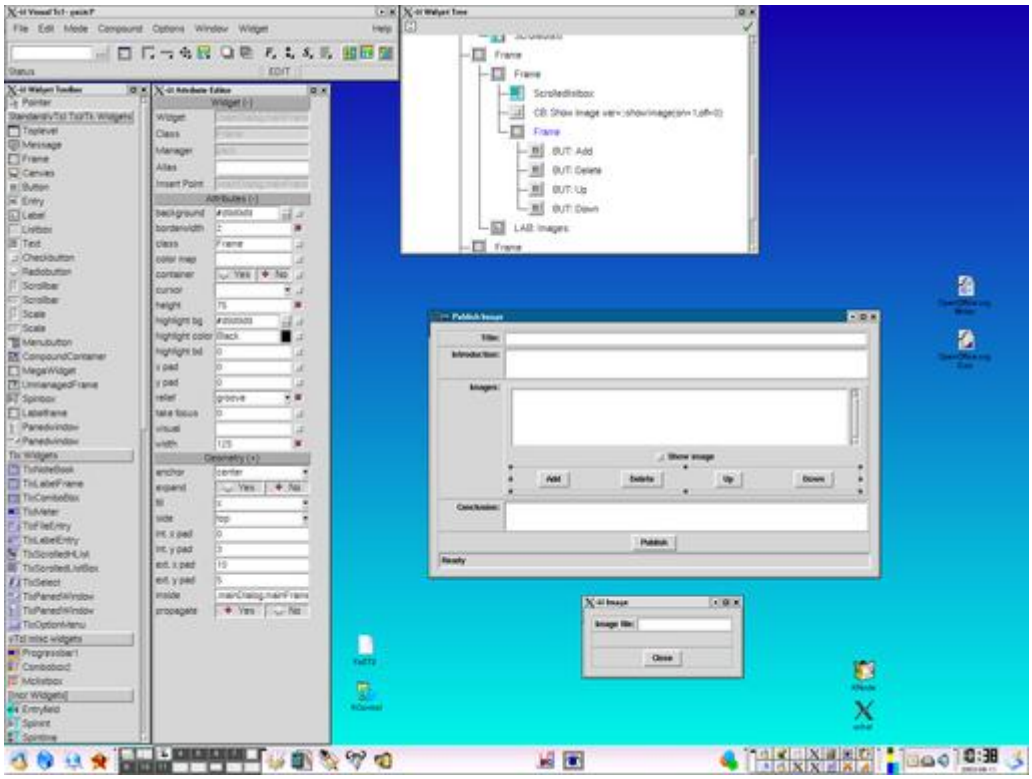


Figure 4. Visual Tcl appears rather cluttered even on a large screen. It's not too hard to use though.

Visual Tcl produces executable Tcl/Tk code, which is loaded and edited directly. The routines that load the Tcl/Tk code are surprisingly tolerant, which means the generated code can be edited and tuned independently by the developer before being returned to Visual Tcl for further work.

Visual Tcl's biggest problem is the dated nature of the toolkit behind it. Tcl/Tk offers only the basic building blocks of widgets. Things like comboboxes and

notebooks aren't available in Tk. Fortunately, a number of extensions to Tcl/Tk provide these mega widgets, and Visual Tcl supports them all. The drawback is that for the final script to run correctly, the target machine needs the mega widget extensions installed. For this project I made use of the incr tcl widget set, and the Tcl/Tk installed as part of most Linux distributions may not contain this set. Hence my recommendation of the ActiveTcl Tcl/Tk distribution. In fact, my SuSE 8.1 system does include incr tcl but strangely doesn't include the extension required to load JPEG images—a rather glaring omission on the part of SuSE I'd have thought.

Anyone who has used a really slick GUI builder tool like the excellent Qt Designer can tell you that Visual Tcl needs more work. It's slow on my dual PIII-500 machine to the point of being irritating, and it has more than its share of usability issues and bugs, although these should be cleared up in the point-one release. The bottom line, though, is Visual Tcl did the job I required. The script it generates is readable enough to be fine-tuned by hand, and anything the code does can be overridden by more specific code in the main application. My GUI completed, I moved on to the application development side of the project.

Building the Application

The thing that sets Tcl apart from more modern GUI scripting solutions is the way the Tk toolkit interacts with the Tcl code that does the work. Packages such as GTK or Qt are low-level libraries, written in C or C++. The script-level bindings to them work well enough, but there's always a big step down from the scripting language into the API of the GUI toolkit. Developers really need to understand the widgets with which they're working and must know how to configure and interrogate them using low-level calls directly to the widgets themselves.

The relationship between Tcl and Tk is much more of a peer-to-peer nature. The GUI toolkit operates at the same level as the language driving it, which makes the combination easy to work with. Take, for example, the listbox widget that contains the list of images to put in the Web page. In Visual Tcl, an attribute of the listbox widget, called the listvar, is presented, and I set it to a variable called ::imageList. ::imageList is a list variable in my Tcl code, and Tcl/Tk ensures that its contents always are reflected in the listbox widget. If I add, move or delete an item in that list variable, the contents of the listbox widget are updated immediately and automatically to display its contents. The code that handles the image list doesn't access or interact with the GUI at all. It simply keeps a single list variable in the correct state, safe in the knowledge that Tcl/Tk does the rest. Figure 5 shows this relationship.

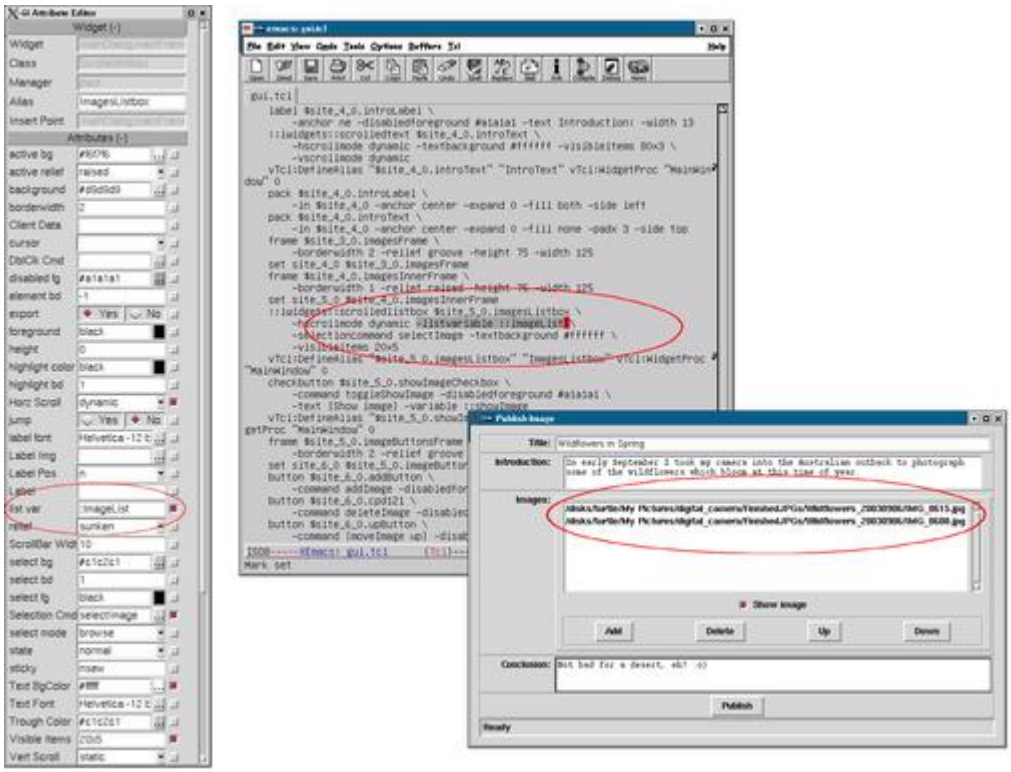


Figure 5. Setting the listvar attribute in Visual Tcl (left) ensures the generated code (middle) causes the onscreen widget (right) to respond immediately to any changes made in the named variable.

More direct access to the widgets sometimes is required. Under these circumstances, Visual Tcl makes use of aliasing. In Tcl/Tk, the name of a widget depends on where it is in the widget tree. That name changes as container widgets, such as frames, are added and removed. To prevent the script writer from having to keep track of the full names of the important widgets, Visual Tcl allows the user to specify an alias—a short, easily memorable name by which the widget always is known. These short names can be looked up in a global associative array (also known as a hash or dictionary), so access to the widgets, wherever they might end up, always is easy. For example, I gave the Introduction text widget the alias IntroText. To fetch the text currently in that widget, the code in Listing 2 can be used.

Listing 2. Fetching the Contents of an Aliased Widget

```
...
set introWidget $::widget(IntroText)
set text [$introWidget get 1.0 end]
...
```

The ::widget array is provided automatically by the Visual Tcl generated code, so fetching the real name of the text widget is simple. Asking the widget to provide its current text, from line 1 character 0 to the end, also is easy.

The image display in the viewer window actually is a label widget in the center of the dialog. Tk can load an image from disk and create a pixmap from it with

one line of code. When the user selects a new image file, a pixmap is created from it and a single command is used to set the label widget to show that image (Listing 3). In the actual script, I store the loaded pixmaps in a cache. This makes switching from one image to another and back again much sharper.

Listing 3. The image is loaded from the disk, and then the label widget is configured to show that image (Tk labels show images as well as text). The image appears on screen immediately.

```
...
set loadedImage [image create photo -file $filename]
$::widget(ImageLabel) configure -image $loadedImage
...
```

When the user clicks the Publish button, a Tcl function is called that creates the Web page. The workings of this code aren't especially relevant here. Suffice it to say that Tcl allows generation of an XML DOM using the TclXML extension and then allows the callout to the libxml2 XSLT processor, which generates the HTML. Getting a specialist package to do the hard work is, of course, the ace up the script writer's sleeve.

The Shortcomings of Tcl/Tk

Although the Tcl/Tk script works nicely, it's hard to ignore the obvious gulf in quality between the appearance of a Tcl/Tk script and a more modern Qt or GTK one. Qt and GTK-based programs look much sharper than those using the Motif style of Tk widgets, plus they are themeable, whereas Tk isn't. Also compare built-in features, such as the file selector dialog—Tk's is no better than GTK's, and both are embarrassed by Qt's. Work continues in the Tcl community regarding these sorts of issues, but as with many mature technologies, improvements are slow in coming for fear of breaking existing code.

Conclusion

Tcl/Tk is the oldest of the GUI-enabled scripting languages in common use today, but it doesn't enjoy the monopoly position it used to have. Python, coupled with GTK or Qt, now provides a more contemporary solution to many of the problems for which Tcl/Tk used to be the natural choice. Both Tcl/Tk and Visual Tcl have some ground to make up in terms of looks, features and desktop integration. Yet, the simplicity of application development offered by the mature and superbly integrated combination of the Tcl language and the Tk toolkit still is second to none. If you have a simple scripting task that would benefit from a GUI, where speed and cost of development are important, Tcl/Tk still should be near the top of the list of contenders for the job.

Resources

ActiveState Tcl Web Site: www.activestate.com/Products/ActiveTcl

Incr Tcl: incrtcl.sourceforge.net/itcl

Practical Programming in Tcl and Tk, 4th edition, by Brent Welch. Prentice-Hall PTR: www.beedub.com/book

The 11 Rules of the Tcl Syntax: www.tcl.tk/man/tcl8.4/TclCmd/Tcl.htm

Source for the Script Developed in This Article: ftp.linuxjournal.com/pub/lj/listings/issue119/7225.tgz

The Tcler's Wiki: mini.net/tcl

Tcl/Tk Headquarters: www.tcl.tk

Tcl/Tk man Pages, On-line and Downloadable: www.tcl.tk/man

Visual Tcl: vtcl.sourceforge.net

XSLT for libxml2: www.xmlsoft.org/XSLT.html

Derek Fountain is a freelance software developer, specializing in UNIX and Linux. He strongly believes in the adage of “make it as simple as possible, but no simpler”. That's why he deploys scripting solutions wherever possible. He lives in Perth, Western Australia.

Archive Index Issue Table of Contents

Advanced search

Building Panoramic Images in The GIMP

Andrew Burton

Issue #119, March 2004

Build big scenic images from your small snapshots using this easy plugin for The GIMP.

Panoramic landscapes make for some amazing photos. There's nothing like the feeling of relaxation and tranquility gained by gazing over the vivid images of sweeping wilderness, minus the hassle of actually getting there. Using a digital camera, it's possible to stitch photos together to simulate the expensive effects of a landscape filter. After I'd bought my digital camera (a Nikon Coolpix 4300) and set it up to work under Linux, getting software to stitch photos together was my next task.

The Nikon Coolpix 4300, like most digital cameras, comes with software on CD to perform rudimentary photo-stitching. Unfortunately, the software is not for Linux. Using Google, it was hard to find anything that would do the job under Linux, until I remembered The GIMP. There are two ways to use The GIMP to create a panoramic photo, easy and hard. The hard way is to set up layers out of the different photos, edit filter and layer masks, mess about with transparency and layer them together, manually.

The easy way is to use Pandora. Pandora is a plugin for The GIMP that takes photos and tries to match the edges of the photos together, using a best guess at where one photo ends and the next begins.

Figure 1. A panorama of suburban Japan. The consistency of the light has made this an easy set of photos to stitch together.

Installation

Because Pandora is a GIMP plugin, to install it, you need The GIMP version 1.2 or 1.3, as well as Gimptool, which is provided in The GIMP development package. Untar Pandora to a working directory, `cd` into it, and run `make`. Pandora detects which version of The GIMP is available and installs it automatically.

Using Pandora

Fire up The GIMP. Pandora should now be available under the Extensions (Xtns) menu as Make Panorama. Select the photos you want to stitch together and click the Add File button; under The GIMP 1.2, you need to add the photos individually, as they should appear from left to right. It's possible to create vertical panoramas, but you need to make use of the rotate feature, as Pandora works horizontally.

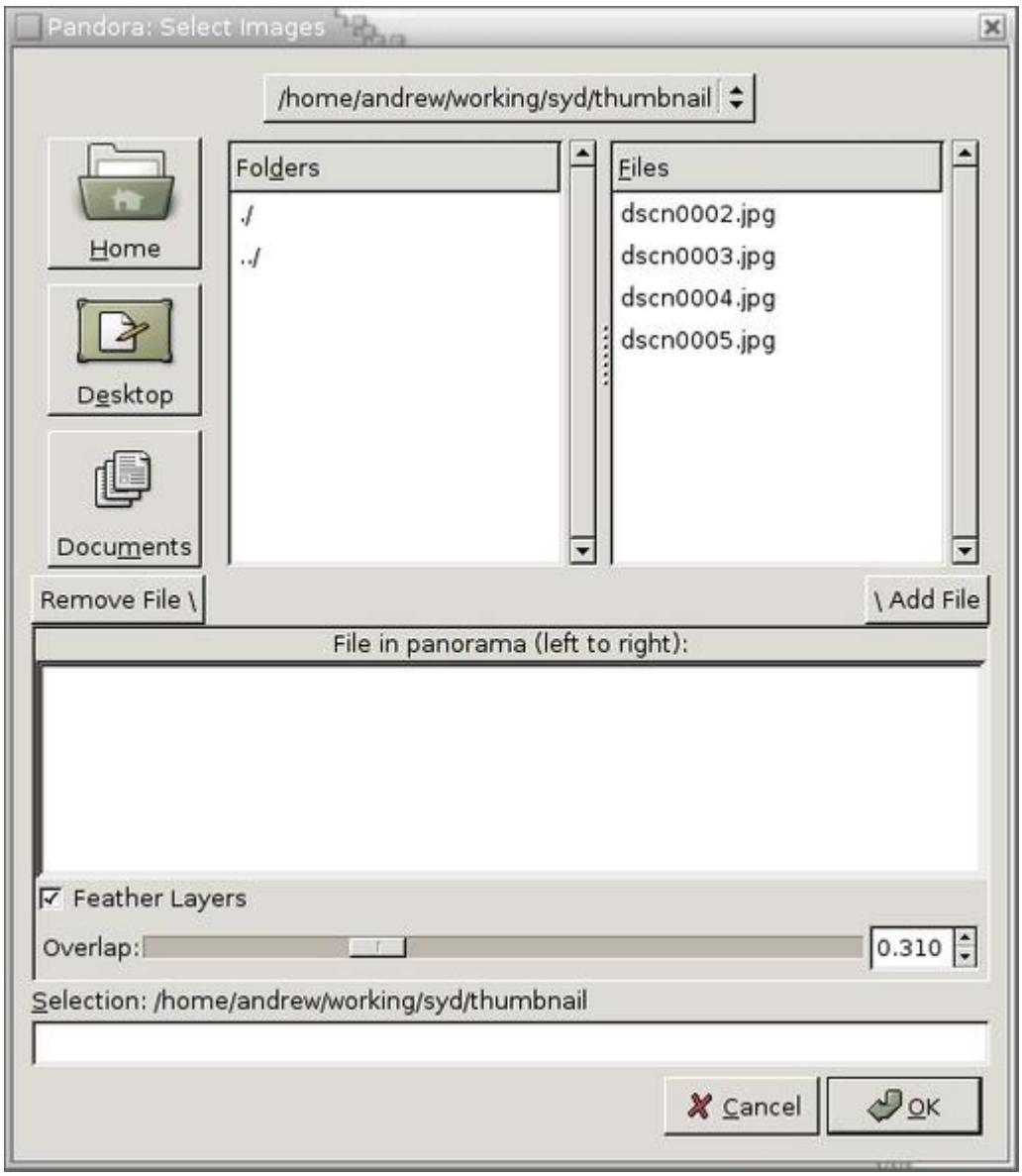


Figure 2. Selecting the Images to Be Incorporated in the Panorama

Pandora can be set with some options before it does its work. The option to feather the layers creates a fade toward the edge of the photos, where the photo becomes slightly translucent. Keep it toggled to create a semi-transparent fade at the sides of each photo, making them easier to line up.

Related to feathering is overlap. Often, photos have minor differences in sky colour; overlap helps to blend the difference so it isn't noticeable. The higher the overlap, the further in from the edge of the photo the feathering takes effect.

Once you're happy with your choices, click OK and Pandora starts to perform its magic.

When the processing has finished, you are presented with a set of layers, one for each original photo. The layers, represented with a dotted line at the edge, should be lined up roughly to what Pandora thinks are the common portions of each picture. Because Pandora is mostly a means of automating the layer creation and feathering, your panorama likely may require a bit more work before you can start impressing your friends.

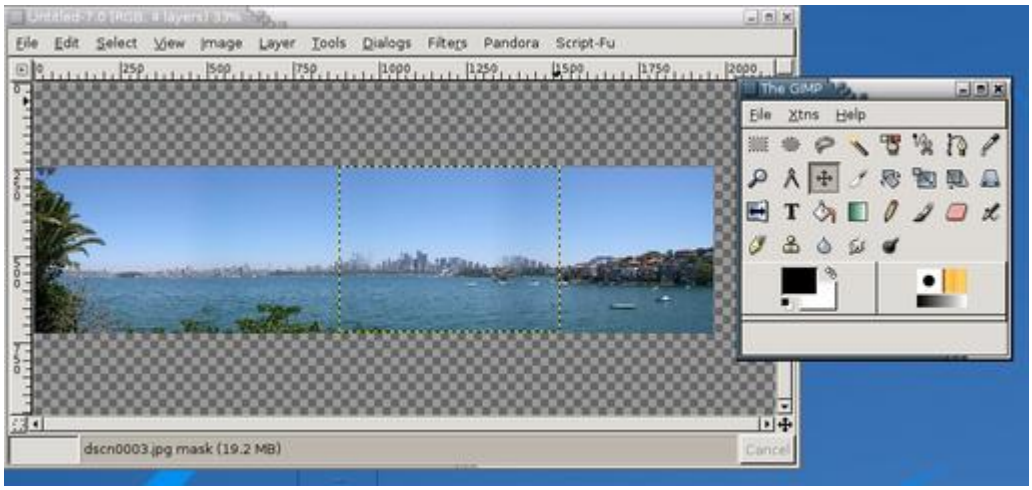


Figure 3. The selected layer is highlighted, and we can move it using the Move layers tool (selected).

Using the Move layers and selections tool (represented by the four-directional arrow), you can select a layer and move it, by holding the left mouse button down while moving the mouse. The easiest way to line the images up is to find a common landmark at the edge of each photo—mountains or trees are ideal—and use these as the anchor around which the images are aligned.

Once the layers are lined up to your satisfaction, you may notice that the pictures have moved out of their perfect vertical alignment, resulting in a jagged top and bottom edge.

Right-click in the image window, and choose Layers→Flatten Image. This merges the layers into one. If you haven't finished lining up the edges, you can undo this last action. Now, using the Select rectangular regions tool, select a region from the bottom left corner to the top right, ignoring all white space caused by the jagged edges mentioned earlier. Copy into a new image, save and you're done.

Your Mileage May Vary

Obviously, Pandora can't cope well in every circumstance. Different amounts of light between photos, particularly when your photos include sky or water, make it difficult to create a consistent picture. This is most notable when shooting toward the sun. Moving subjects, such as cars or people, can result in the occurrence of ghosted images. Cityscapes containing a lot of right angles can emerge imperfectly when the angle of each photo is not perfect. Finally, if the source photos are not ideal, your results won't be either. A fixed tripod or at least holding the camera close to you with your elbows against your body gives a standard height and angle that can make your photos much easier to line up. The better your source photos, the less effort you need to use in making your panoramas fantastic.

As with most things, you can find tutorials and hints on creating panoramas on the Net. By using Pandora, it's possible for a rank amateur to come up with some great results, even with a limited knowledge of The GIMP and layers. The picture in Figure 4 shows a successful scene, where the sky and water tones are consistent and the edges are lined up.

Figure 4. Sydney from Cremorne Point—water and buildings are difficult, but the rewards are worth it.

Resources

Pandora is a small download from the Shallow Sky Web site: www.shallowsky.com/software/pandora.

Panoguide is the definitive resource for panoramic photos: www.panoguide.com.

Red Skies at Night (previously mentioned in *LJ*, April 2003) has some great GIMP tutorials for digital photography enthusiasts: cs.uhh.hawaii.edu/~jeschke/photography/articles/gimp/tutorials.shtml.

Andrew Burton (adb@iinet.net.au) lives in Sydney, Australia, where there is plenty of inspiration for taking panoramic photos.

Archive Index Issue Table of Contents

Advanced search

Designing Tip Windows

Hugh Fisher

Issue #119, March 2004

If you're itching to teach users how to use your application, make a tip window they won't want to turn off.

The first thing you see in many desktop applications is the “Tip of the Day” or “Did You Know?” window. I've seen an increasing number of tip windows on recent Linux systems, so here is a discussion of the most effective way to implement them. A tip window is not the same as a splash screen, which is the window that is briefly displayed while a program loads, links, configures and generally gets itself ready. The program closes the splash screen itself without any user action.

Even splash screens have variations. Usually they are created by desktop applications, but bootloaders and graphic drivers, such as NVIDIA, also can have them. Microsoft Office products typically have splash screens full of intimidating legal warnings. Adobe does nice splash screens: the legalese is low-key, there's a nice piece of artwork and a small text field shows the names of extensions and plugins as they are loaded. If you have an attractive splash screen for your application, document somewhere the location of the PNG logo it displays. You may be lucky and the user will put it on a Web page.

Why have splash screens? In an ideal world, we wouldn't. You can go a long way as a user interface designer by remembering only a few rules, among them the magic numbers 100ms and two seconds.

For most purposes, anything the computer can do in 100ms or less is perceived as instantaneous. If you are writing the pilot training simulator for a jet fighter, you have to be more precise, so if you can get your application to launch in 100ms, you've done a fabulous job, users will love your product, and a splash screen would be superfluous or might even be considered an illegal subliminal image.

The two-second limit is, on average, how long the computer can keep users waiting before breaking their concentration and making them aware that their time is being wasted by a machine. A launch time under two seconds ought to be possible for most desktop applications, but sometimes it is out of the author's control. The splash screen is meant to hide the delay.

A tip window is different. It tries to be useful rather than merely decorative, and it has to be dismissed by the user. It doesn't vanish of its own accord. The introductory sequences in many games are tip windows. Visually, they look quite different, but the function is exactly the same. You watch the briefing or backstory or whatever until you've learned what you need and click to continue.

Tip windows have become common in recent Linux distributions, matching Macintosh and Windows environments. And, as in the Mac and Windows worlds, 99 out of 100 users click the Don't Show Again button within a few days and never look at the tips again. This is a shame, because tip windows really are a good idea. We all know nobody reads manuals. A tip window gives you, the application developer a chance to walk the user gently through the capabilities of the application, presenting information in small convenient chunks. It doesn't even cost users any time; they have to sit through the launch delay anyway.

How can we encourage users not to turn off the tip window? Well first, why do they? Here it's useful to discover what is going on with a GOMS keystroke model. (GOMS—goals, objects, methods, selection—is a way of analyzing user interfaces and interaction.) Applied to tip windows, the GOMS keystroke model shows that the tip window has introduced a second unnecessary action to the launch process.

Taking a word processor or text editor as our example, the user's goal is to write something. The action is to click or double-click the appropriate icon. With no tip window, only a splash screen, no further action is required and users can start typing as soon as the application launches. The tip window, though, forces users to carry out a second action to dismiss it. The annoyance of this extra action is why tip windows are turned off; it has nothing to do with the helpfulness of the content.

If you're not convinced that merely one extra click can make such a difference, consider that “nagware” in the Mac/Windows environments relies exactly on this behavior. These applications are shareware and require a license fee but are free to download. Every time the application launches it shows a window reminding that you haven't paid yet. You have to dismiss this window every time. Only after you pay will the author send you a code that disables the nag window. It works because it is annoying.

Turning the tip window back into a splash screen and closing it as soon as the application has launched would remove the annoyance. However, only speed readers would be able to absorb the tips, which rather defeats the purpose. A fixed delay of a few seconds would annoy people in a hurry. The right thing to do is to close the tip window as a side effect of the user's first action. More technically, the first mouse entry, motion, button event or key event, closes the tip window and is then processed as normal by the application. Now the user can pause to read the tip window if it looks interesting or simply start working if not.

This isn't an original idea, by the way. Start a copy of Emacs or xemacs with no filename given. You get a blurb about Emacs, the Free Software Foundation and how to get more information, but the first key you press clears it all and is inserted into a new document. Perfect.

There is one small new problem: if the tip of the day is particularly fascinating, how can the user save it? They can't copy the text, because whatever they try will close the tip window. So, the application should remember which tip was displayed at launch and set the on-line help system to always open with that same text as the initial contents.

Resources

For the full GOMS model: Card, Stuart K., Moran, Thomas P., Newell, Alan. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, 1983.

For the useful keystroke level subset, plus many interesting and provocative ideas: Raskin, Jef. *The Humane Interface*, ACM Press, 2000.

Hugh Fisher (hugh.fisher@anu.edu.au) is a system administrator, 2-D/3-D interactive graphical programmer and part-time lecturer. He has strong opinions on the usability of Linux systems and hopes to inflict these on a wider audience by writing for *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Fast Convenient Mail for Travel: OfflineIMAP

John Goerzen

Issue #119, March 2004

Linux laptop users, try the mail solution that combines the advantages of fast local mail folders and a server-based IMAP repository.

E-mail is for many people the single-most important feature of the Internet. We read e-mail at home, at work, while traveling and on many different computers. But it's difficult to see the same mail from all of those places. If you delete a message from home, it may not show up as deleted when you look at the same account from work. Worse, you might be able to view a given message on only one machine. And if you sometimes want to download mail to your laptop and read it without any Internet connection, things get even more complex.

Some people try to solve these problems by using IMAP in their mail clients. But IMAP can be slow and poorly supported; especially on a slow connection, it tends to make mail reading unpleasant. I recently faced exactly this situation—I was a very annoyed programmer. Many programs come about because a programmer somewhere was annoyed. Thus, I wrote OfflineIMAP.

About OfflineIMAP

OfflineIMAP is designed to let you read the same mailbox from many different computers, with or without an active Internet connection. It performs a bi-directional sync, which means that any changes you make eventually are reflected on all your machines. In its most common form, OfflineIMAP works by connecting to an IMAP server and synchronizing your folders to a series of Maildir folders on your local machine. Despite its name, OfflineIMAP is useful even if you never read mail off-line.

Installing OfflineIMAP

OfflineIMAP installation is easy. Visit the OfflineIMAP home page at quux.org/devel/offlineimap, and download the .deb or the tar.gz file. Debian users simply

can run `dpkg -i offlineimap.deb` to install it, and then use `apt-get -f install` to fix any missing dependencies. If you're not running Debian, make sure you have Python 2.2 or above installed. If you do not have Python already, check with your distribution or visit www.python.org to download it.

When you're ready to install OfflineIMAP, run `tar -zxvf offlineimap_4.0.2.tar.gz` to unpack the source. Change into the new directory and then, as root, run `python setup.py install`. If you get stuck, the OfflineIMAP manual contains some more installation hints.

Basic Configuration

OfflineIMAP configuration is done in the ~/.offlineimaprc file. That file has three different sections: general, which controls overall behavior of OfflineIMAP; repository, which describes a place where mail is stored; and account, which describes how two repositories are synchronized together. A basic, simple setup requires only a small configuration file. Here is an example:

```
[general]
accounts = MyMail

[Account MyMail]
localrepository = MyMailLocal
remoterepository = MyMailRemote

[Repository MyMailLocal]
type = Maildir
localfolders = ~/MyMail

[Repository MyMailRemote]
type = IMAP
remotehost = hostname.example.com
remoteuser = my-username-goes-here
ssl = yes
```

This example defines one account, MyMail. The MyMail account is synchronized from the hostname.example.com server to the ~/MyMail directory on your local machine. All remote folders are copied. If your IMAP provider does not support SSL encryption, delete the `ssl = yes` line. Now, run `offlineimap`. You are asked for your password, and then it synchronizes your mailboxes once and exits.

Continuous Synchronization

If you're connected to the Internet while you read your mail, you can have OfflineIMAP continually keep your local tree synced up with the server. To do this, simply add an `autorefresh` line to your account section. For instance, you might modify your account section to look like this:

```
[Account MyMail]
localrepository = MyMailLocal
remoterepository = MyMailRemote
autorefresh = 5
```

When you run OfflineIMAP now, it synchronizes your mailbox like before. But when it's done, instead of exiting, it keeps running, synchronizing your mail every five minutes.

Synchronizing Multiple Accounts

OfflineIMAP is quite capable of synchronizing multiple accounts. For instance, you might want to be able to read mail from both your work e-mail and your home e-mail. To do this, add one account and two repository sections for each account, making sure to use unique names. Then, add the account to the accounts list in the general section. Separate the names by commas.

On the local side, you should make sure that each account synchronizes into a different directory. Otherwise, confusion and corruption may occur.

Boosting Performance

OfflineIMAP's defaults, as illustrated with the examples above, are quite conservative. It tries to work with as many IMAP servers as possible right out of the box, so the advanced features that occasionally cause trouble are disabled by default.

If you have many mail folders or get a lot of mail in each folder, the synchronization process can be slow. This is especially true if you are using a high-latency Internet connection, such as a modem or satellite. To speed things up, OfflineIMAP is capable of establishing multiple connections to your server at once. It then is able to perform tasks in parallel. For instance, OfflineIMAP might download three messages and synchronize two folders simultaneously.

OfflineIMAP offers several configuration options. First, you should add a line such as `maxsyncaccounts = 5` to your general section. This enables OfflineIMAP to synchronize multiple accounts simultaneously, which is almost always a good thing. Second, in the repository section for the remote part of each account, you can control how much parallelism to use. For instance, you might add a line saying `maxconnections = 3` to the MyMailRemote repository section in our example. This allows OfflineIMAP to establish up to three connections to the server.

If you are performing continuous syncs with the autorefresh option described above, there's another source for delay. Each time OfflineIMAP starts syncing an account, it connects to the server. When it's done with that particular sync, it disconnects. Establishing these connections can be slow in many cases. OfflineIMAP provides an option to keep the connections open even between syncs. The problem is that some servers disconnect clients that are idle for a long time. To combat that problem, OfflineIMAP also can send little bits of

traffic every so often to make sure the timers don't expire. To take advantage of these features, add lines like these to the remote repository section:

```
holdconnectionopen = true
keepalive = 60
```

Keepalive is given in seconds, whereas autorefresh is given in minutes.

User Interfaces

OfflineIMAP ships with many different user interfaces. The two most common are Tk.Blinkenlights and Curses.Blinkenlights. The former presents a small graphical window on OfflineIMAP's progress on your X desktop. The latter runs in a terminal and provides a nice monitor of progress (see Figures 1 and 2).

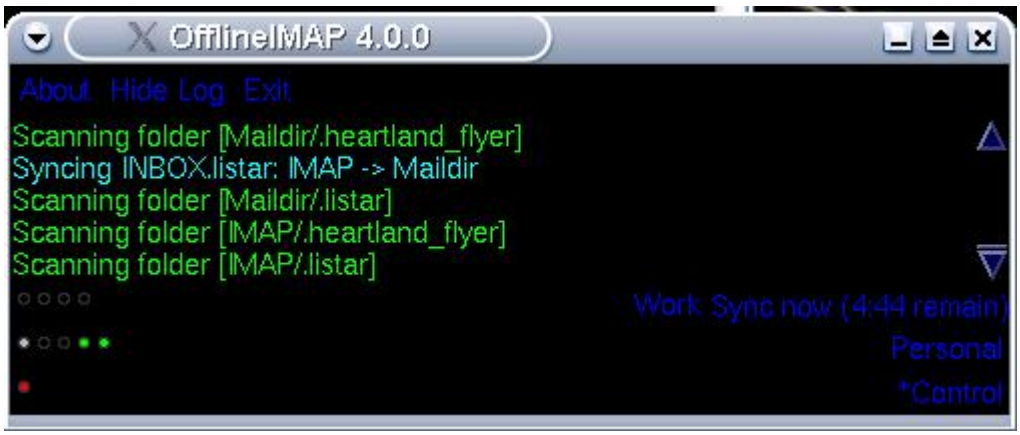


Figure 1. The Tk.Blinkenlights GUI Interface for OfflineIMAP

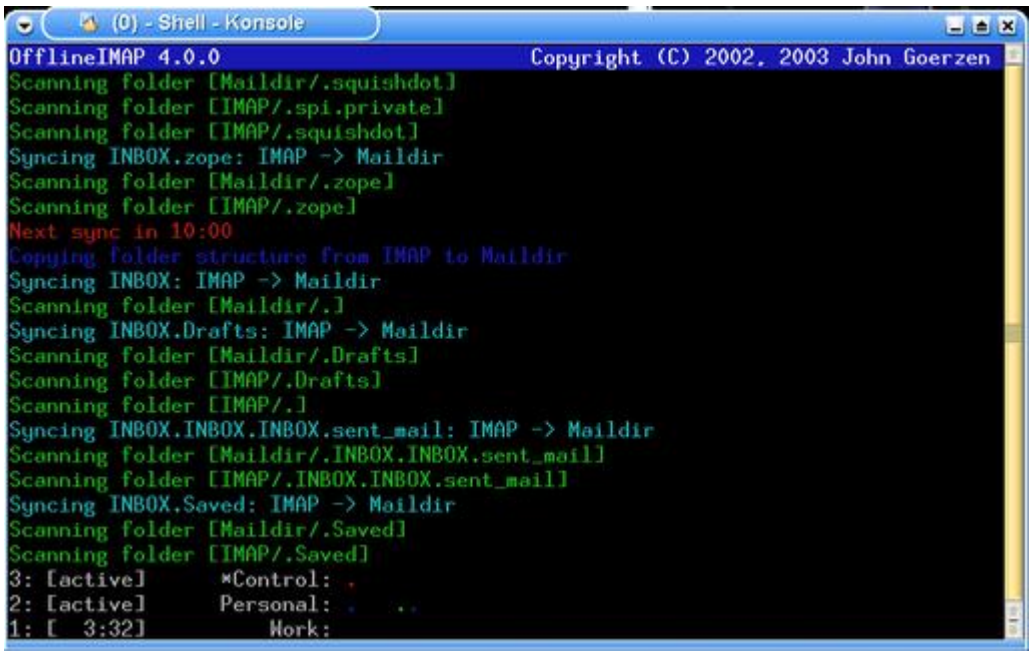


Figure 2. The Curses.Blinkenlights Interface, Running in a Terminal Window

With the Tk.Blinkenlights interface, you can click on the Sync immediately button to force the account to synchronize right away. You can do the same thing in the Curses.Blinkenlights interface by pressing the number next to the account name. Both interfaces display a log of current activities. You also get a mesmerizing display of flashing status lights so you won't get bored while watching the synchronization happen.

The TTY.TTYUI interface can run without any Curses support—it uses no color or terminal controls. It can read password input, but it provides no other capabilities for interaction.

Noninteractive.Basic is a user interface designed never to receive any input from the user. It can, however, display status messages. If you need a password in order to log in to a remote server, add a line such as `remotepass = mypassword` to the remote repository section of the configuration file.

Finally, Noninteractive.Quiet goes one step further and does not output status messages. Some people like to run OfflineIMAP from a cron job, and Noninteractive.Quiet is good for that.

You can specify which user interface should be used in one of two ways. First, you can use the -u option on the OfflineIMAP command line. For instance, you might run `offlineimap -u Curses.Blinkenlights`. Alternatively, you can add a ui line to your general section, like this:

```
ui = Tk.Blinkenlights, Curses.Blinkenlights,
     TTY.TTYUI
```

With this configuration, OfflineIMAP first tries the Tk.Blinkenlights interface. If your Python doesn't support Tk, or if you are not running under X, it then tries the Curses.Blinkenlights interface. If that too fails, the TTY.TTYUI interface is tried. If even that does not work, OfflineIMAP aborts with an error.

Selecting Folders

By default, OfflineIMAP asks your remote IMAP server which folders are available to you and synchronizes all of them. A folderfilter option can be added to your remote repository section to restrict what is brought over. The folderfilter option is a tremendously powerful option. Unlike the other options you've seen so far, folderfilter actually expects to be handed a Python function. The function takes one argument and should return true if that folder is to be included.

Python provides a feature called lambda that lets you create simple functions on the fly. You thus can construct some complex rules. Here are a few

examples. You can specify a set of folders you want to synchronize. You can use the Python in operator to test whether the folder in question is in the list, like this:

```
folderfilter = lambda foldername: foldername in
      ['INBOX', 'Sent Mail',
       'Received']
```

This code synchronizes only the three named folders. Notice the indentation on the second and third lines—if you indent them, the configuration parser treats them as part of a single statement.

You also can specify folders to exclude:

```
folderfilter = lambda foldername: foldername not in
      ['Spam', 'Junk']
```

In this example, all folders except Spam and Junk are synchronized.

You also can use regular expressions, such as:

```
folderfilter = lambda foldername:
     not re.search('(^Trash$|Del)', foldername)
```

This input causes the folder named Trash and all folders containing the text Del to be excluded.

Changing Folder Names

Sometimes, you may want to alter the folder names before storing the folders on the local side. OfflineIMAP provides an option called nametrans, also specified in the remote repository section, to do exactly that. Some IMAP servers, such as Courier, add “INBOX.” to the start of all folders, which can be annoying. The nametrans feature lets you get rid of that. Here's an example:

```
nametrans = lambda foldername:
   re.sub('^INBOX\.', '', foldername)
```

Like folderfilter, nametrans takes a Python expression. This expression receives a folder name as an argument and should return the new and improved folder name. In this example, any folder whose name starts with INBOX. gets the leading INBOX. removed. It's important to remove not only the leading INBOX; the folder INBOX itself does exist, so you'd wind up with an empty folder name (and that's a bad thing).

It's also important to be careful with your nametrans rules. You must make sure that nametrans returns a different value for each folder. If it returns the same thing for two different folders, bad things can happen.

In case you're wondering, nametrans does not change folderfilter. That is, your folderfilter rules operate on the folder names before nametrans takes effect.

Synchronizing Two IMAP Servers

Some mail readers don't support Maildir hierarchies well. For them, OfflineIMAP introduced a new feature: the ability to synchronize two IMAP servers directly. The idea is this: you install an IMAP server on your local machines. Your mail reader, which already may have slow IMAP support, is fairly speedy in accessing an IMAP server located on your own machine. The mail reader never needs to know that OfflineIMAP is sticking the messages in the folders.

To make this happen, you need to make a few simple changes to your local repository section. First, change the type from Maildir to IMAP. Secondly, remove the localfolders and other Maildir information and instead specify IMAP configurations, such as remotehost and remoteuser. Finally, delete your ~/.offlineimap directory to make sure that none of the old status information lingers around.

Certain options still are supported only in the remote section—nametrans and folderfilter are two examples—but the options relating to the connection itself are supported in both places. You can, in fact, have your local IMAP server on a machine that is remote to you.

Conclusion

OfflineIMAP is a powerful mail solution. I've introduced you to the basics of OfflineIMAP in this article, but there still is more you can do with it. To learn more, check out the OfflineIMAP home page and the example configuration files. If you're a Python programmer, you'll find some nice hooks for Python code as well.

John Goerzen has been programming for Linux since 1996 and currently is vice president of Software in the Public Interest, Inc. He welcomes your comments at jgoerzen@complete.org.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Power Management in Linux-Based Systems

Srivatsa Vaddagiri

Anand K. Santhanam

Vijay Sukthankar

Murali Iyer

Issue #119, March 2004

Implementing power management in any system is a complex task. Here's how to manage your system's transitions from normal run state to power-saving modes.

Power management (PM) software is a crucial component in battery-powered systems, such as PDAs and laptops, because it helps conserve power when the system is inactive. As a simple example, power may be conserved by switching off the display when a system is inactive for some time. Conserving power in this manner extends battery life, so one can work more hours before having to recharge the battery.

Hardware support is vital for power management to work, and software intelligently exercises that support. The degree of power management support available in hardware varies from device to device. Some devices, such as a display, simply provide two power states, on and off. Other devices, like the SA1110 CPU, may support more complex power-saving features, including frequency scaling.

Implementing power management in any system is a complex task, considering that several non-interacting subsystems need to be brought together under a single set of guidelines. This article explains how power management works in Linux (2.4.x) and how it can be implemented in battery-powered systems based on an APM standard, at both the device driver and application levels.

Two Power Management Standards

Power management for computer systems has matured over the years and several standards exist. The two popular ones are advanced power management (APM) and advanced configuration and power interface (ACPI). APM is a standard proposed by Microsoft and Intel for system power management, and it consists of one or more layers of software to support power management. It standardizes the information flow across those layers. In the APM model, BIOS plays a key role. ACPI is the newer of the two technologies, and it is a specification by Toshiba, Intel and Microsoft for defining power management standards. ACPI allows for more intelligent power management, as it is managed more by the OS than by the BIOS. Although both standards are more popular in x86-based systems, it is possible to implement them in other architectures.

Power Management Implementation

Before implementing power management, it is important to understand what hardware support is available for saving power. One of the important goals of power management software is to keep all devices in their low power states as much as possible.

A possible approach for implementing power management is first to define a power state transition diagram. This defines several power states for the system and also defines the rules and events governing state transitions.

As an example, consider a PDA that has the following devices: Intel SA1110 CPU, real-time clock, DRAM, Flash, LCD, front light, UART, audio codec, touchscreen, keys and power button. The Intel SA1110 CPU supports several power-saving features, including frequency scaling, where the core clock frequency can be configured by software. Lowering clock frequency reduces the CPU's power consumption, but at the cost of reduced CPU speed. This CPU also supports several modes of operation:

- Run mode: the normal state of operation for the SA1110 when it is executing code. All power supplies are enabled, all clocks are running and every on-chip resource is functional.
- Idle mode: allows software to stop the CPU when not in use. In this mode, the CPU clock is stopped, representing some savings in power. All other on-chip resources are active. When an interrupt occurs, the CPU is reactivated.
- Sleep mode: offers the greatest power savings and consequently the lowest level of available functionality. In this mode, power is switched off

to the majority of the processor. Some preprogrammed event, such as a power button press, wakes up the CPU from this mode.

As you can see, software is responsible for transitioning the CPU either to idle mode or sleep mode.

In such a PDA, DRAM cells normally are refreshed periodically by the memory controller logic present inside the CPU. In sleep state, however, the majority of the CPU is shut off, which results in DRAM cells not being refreshed, which in turn leads to loss of data in DRAM. To avoid this loss, most DRAMs support a mode called self-refresh wherein the DRAM itself takes care of refreshing its cells. In such cases, software can put DRAM in its self-refresh mode by writing to a few control registers before transitioning the CPU to its sleep mode, thereby preserving the DRAM contents.

The top power-hungry devices in this PDA can be the CPU, DRAM and display back light. Hence, they should be kept in their low power states as much as possible.

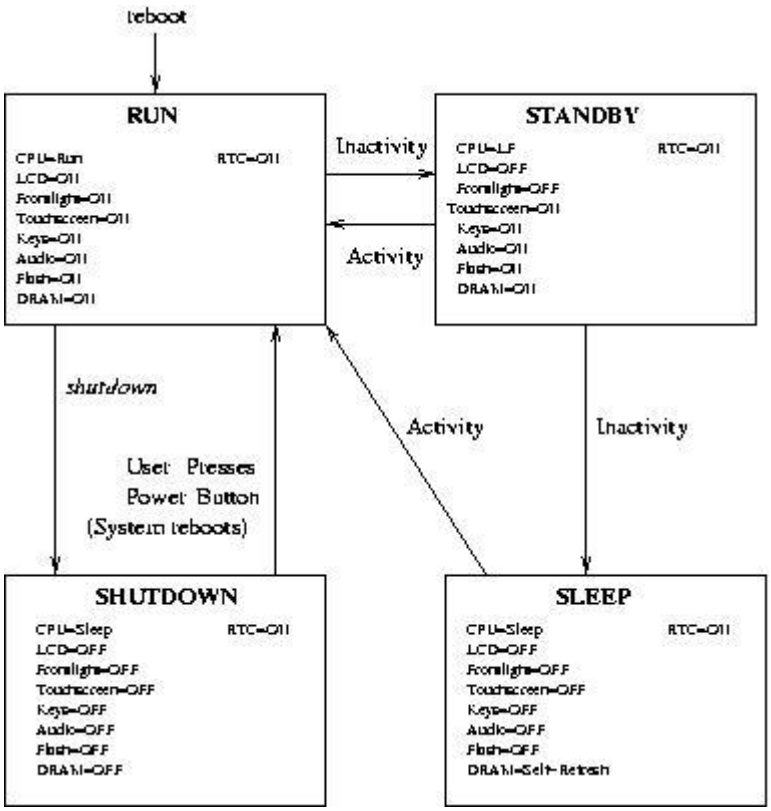


Figure 1. Power State Transition Diagram

Figure 1 shows a possible power state transition diagram for this PDA. Here is a brief description of the power states:

- Run state: system falls into this default state when it reboots. Power consumption is maximum in this state, as all devices are turned on or active.
- Standby state: system falls into this state due to inactivity. LCD and display back light are turned off, and CPU clock speed is reduced to save some power.
- Sleep state: system falls into this state due to continued inactivity. Power is conserved aggressively by putting the CPU in sleep mode, which in turn powers off most devices. DRAM, however, is put in its self-refresh mode to preserve the machine state (system and application text/data loaded in memory) while the system is sleeping. The system awakens from sleep state when a preprogrammed event occurs. When it wakes up, it transitions to the run state and machine state is restored.
- Shutdown state: system falls into this state when the shutdown command is issued. The system reboots when it exits from this state. This means it is not necessary to preserve the machine state in DRAM, and hence DRAM can be powered off. The shutdown state then represents the lowest power consumption state of all.

The real-time clock is kept on in all power states to retain system time.

It is clear from this diagram that detecting inactivity and putting the devices in their low power states forms the heart of power management software.

Linux and Power Management

Power management software manages state transitions in association with device drivers and applications. It intimates all PM events, including standby transition, sleep transition and low battery, when they occur. This allows software to veto certain state transitions when it is not safe to do so.

Device drivers generally are responsible for saving device states before putting them into their low power states and also for restoring the device state when the system becomes active.

Generally, applications are not involved in power management state transitions. A few specialized applications, which deal directly with some

devices, may want to participate. This section explains what device drivers need to do in order to participate in power management:

- pm_dev structure: the PM subsystem in the Linux kernel maintains some information in a pm_dev structure about every registered driver. Maintaining this information allows it to notify all registered drivers about PM events.
- pm_register: device drivers first have to register themselves with the PM subsystem before participating in power management. They do this by calling pm_register:

  ```
  struct pm_dev *pm_register(pm_dev_t type, unsigned
  long id, pm_callback cbackfn);
  ```

  where *type* is the type of device being managed by the driver, *id* is the device ID and *cbackfn* is a pointer to some function in device driver. This is called as the driver's callback function.

The linux/pm.h file defines the various types and IDs that can be used by drivers. If successful, pm_register returns a pointer to a structure of type pm_dev. A driver's callback function is invoked by the PM subsystem whenever there is a PM event. The following arguments are passed to the function:

- dev: a pointer to the pm_dev structure that represents the device; the same pointer returned by pm_register.
- event: identifies the PM event type. The possible events are PM_STANDBY, meaning the system is going into standby state; PM_SUSPEND, meaning the system is going into suspend state; and PM_RESUME, meaning the system is resuming (from either standby or sleep states). Based on implementation, more events can be supplied.
- data: data, if any, associated with the request.

Each device driver is supposed to do some processing according to the PM event type. In a PM_SUSPEND event, for example, the LCD driver is supposed to save the device state and then switch off the LCD. If it is a PM_RESUME event, the LCD driver should switch on the LCD and restore its state from the saved state.

The callback function should return an integer value. Returning a value of zero signifies that the driver agrees to the PM event. A nonzero value signifies that the driver does not agree to the PM event. This may cause the state transition in progress to be aborted. For example, if a PM_SUSPEND event is sent to the LCD driver's PM callback function and it returns 1, the suspend operation is aborted.

All the driver's callback functions are invoked in a predefined order. This is on a last-come-first-served basis, which can be a problem if two devices depend on each other. Let's say the interface to a Bluetooth (BT) device is through a USB host controller (HC). The Bluetooth driver needs this interface to be up before it can talk to the BT device. Because of this dependency, the USB HC driver is loaded before the BT driver. This means the USB HC driver registers with PM before the BT driver.

Whenever a system wants to transition to sleep state, a PM_SUSPEND request is sent first to the BT driver and then to the USB driver. The USB HC driver may shut off the BT port as part of its PM_SUSPEND processing. When the system resumes, PM_RESUME is sent first to the BT driver and then to the USB HC driver. At the time when the BT driver processes this request, its interface to the BT device is not available, and hence it may have problems in resuming the BT device. One way of tackling this situation is to change the PM_RESUME order in the kernel to be on a first-come-first-served basis.

A driver stops participating in power management by calling pm_unregister:

```
pm_unregister(pm_callback cbackfn);
```

To unregister, it has to supply the pointer to the same function it used while registering. Once a driver has unregistered itself, the PM subsystem stops involving it in further PM events.

Linux also defines two interfaces, pm_access and pm_dev_idle, for drivers; pm_access should be called before accessing hardware, and pm_dev_idle has to be called when the device is not being used. These interfaces cannot be implemented on all platforms, though.

Now we illustrate how a typical state transition takes place when only device drivers are involved. The PM subsystem maintains all drivers that have registered with it in a doubly linked circular list. Figure 2 shows how this list looks when three drivers, A, B and C, have registered with it. This assumes that driver C registers first, then B and finally A.

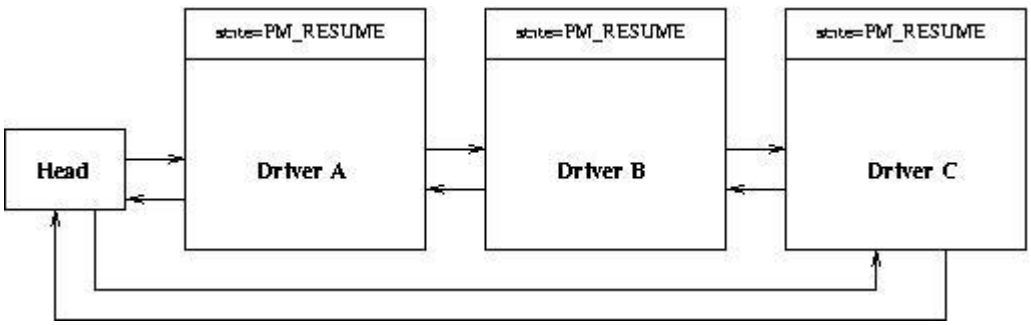


Figure 2. System in Run State

Now, let's say the system has to transition to standby state from run state. PM subsystem sends out a PM_STANDBY request to all three drivers, for which there are two possible outcomes. One, all drivers accept the request, and the system is put in standby state. Two, some driver rejects it. In this case, the standby transition is aborted, and the system continues to be in run state.

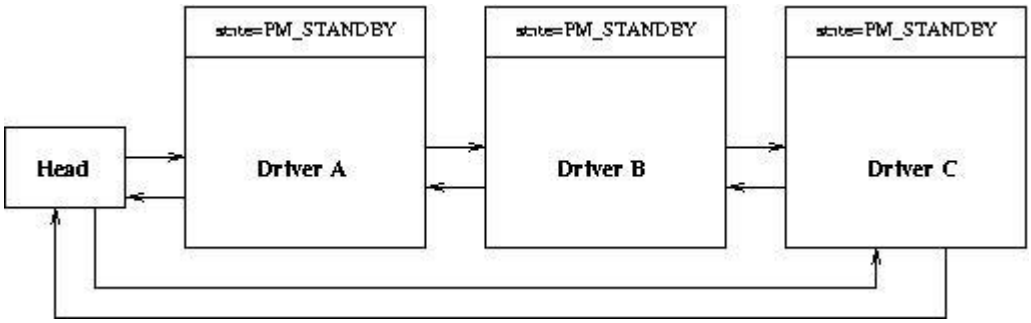


Figure 3. System in Standby State

Figure 3 shows what happens when all the drivers have accepted the PM_STANDBY request. Notice how the state field in pm_dev structure is changed when a driver accepts the request.

Let's now consider the case where drivers A and B accept the PM_STANDBY request, but driver C rejects it. Figure 4 shows the case after driver A has accepted the request. After driver A has agreed, the PM_STANDBY request is sent to driver B.

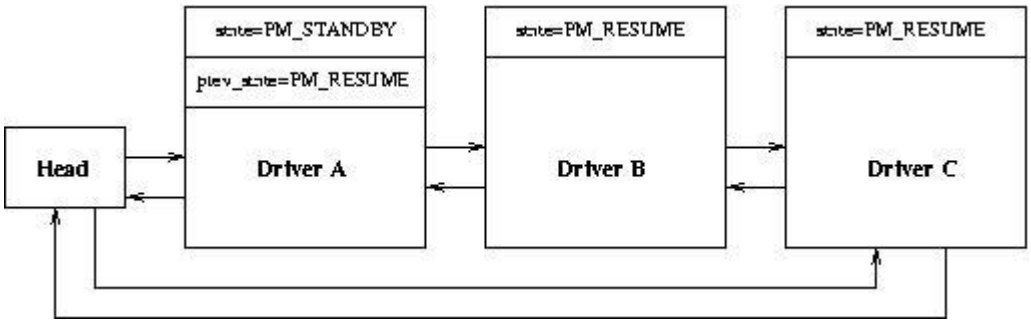


Figure 4. Driver A has accepted the PM_STANDBY request.

Figure 5 shows the state of the drivers after driver B also has accepted. Now both devices A and B are put in their standby state, while device C is still in its run state.

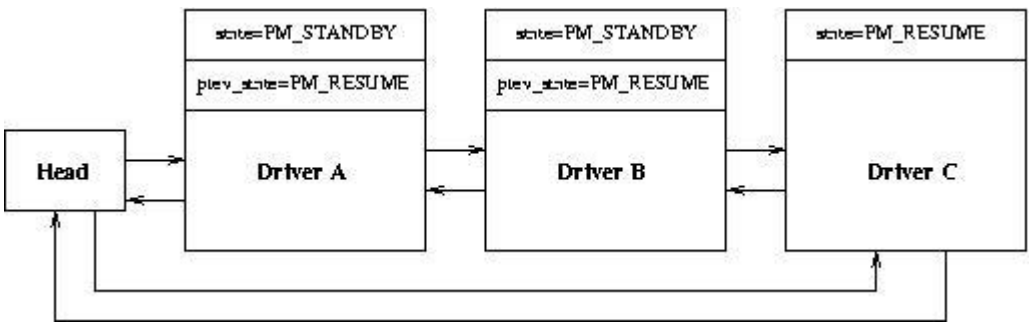


Figure 5. Driver A and driver C have accepted a PM_STANDBY request.

Next, PM_STANDBY is sent to driver C, which rejects it. In this case, the standby transition has to be aborted. Because devices A and B already have been put in their standby states, the PM subsystem has to perform an undo operation on them, so it sends a PM_RESUME request first to driver B and then to driver A. After this undo operation is done, all devices are put back in their run states, as shown in Figure 6.

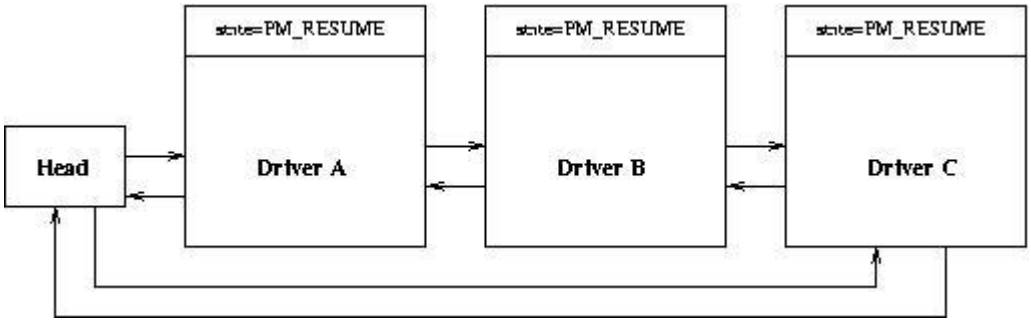


Figure 6. The system is back in run state after driver C rejected the PM_STANDBY request.

APM

Figure 7 shows the APM model. The important components of this model are:

- APM BIOS: software interface to the motherboard and its power managed devices and components. It is the lowest level of PM software in the system.
- APM driver: implements APM in a particular operating system.
- APM-aware device drivers and applications: APM driver interacts with them for all PM events.

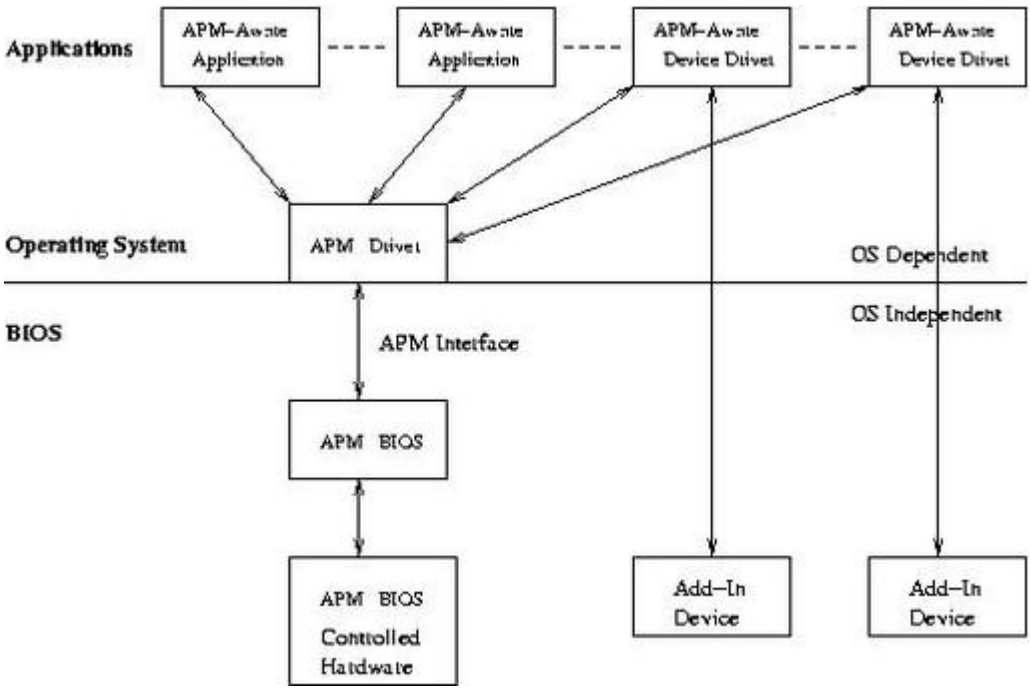


Figure 7. APM Model

APM BIOS detects and reports various PM events, including low battery, power status change, system standby, system resume and so on. The APM driver uses polling function calls to the APM BIOS to gather information about PM events. It then processes these events in association with APM-aware drivers and applications.

The APM driver in Linux exposes two interfaces for an application's use. The first, /proc/apm, holds information on the system power. It specifies whether the system is running on A/C power or battery. If running on battery, it also specifies the battery charge and time left for the battery to drain completely. The second interface, /dev/apm-bios, allows applications to know of and participate in PM events. It also allows them to initiate power state transitions by themselves, by issuing suitable ioctl calls. Read calls issued against this file will block until the next PM event occurs. When the read call returns, it carries information regarding the PM event about to occur.

Some of the applications that have opened /dev/apm_bios may be running with root privileges. Such applications are special to the APM driver. For some of the events, such as standby or suspend transition, APM driver informs all applications that have opened /dev/apm_bios about the event. In addition, it waits for approval from those few applications running with root privileges before the system actually is put in standby/suspend state. This approval comes when applications issue suitable ioctls.

The following ioctls normally are supported:

- APM_IOC_STANDBY: puts the system in standby state.
- APM_IOC_SUSPEND: puts the system in suspend state.

APM also comes with two user-space utilities. The apm command interacts with the APM subsystem in the kernel. Depending on the arguments passed, it can display system power status, or it can be used to initiate system standby/ suspend transition. The apmd dæmon reports and processes various PM events and logs all PM events to /var/log/messages. In addition to logging, apmd also can take some specific actions for each type of PM event. These actions are specified in a script file (usually called apmd_proxy). This script file is invoked by the apmd dæmon with one or two arguments indicating the PM event about to occur. The following is a sample script file:

```
case 1:2 in

"standby":*)
     #System is going to Standby state because of
     #inactivity. Reduce CPU speed.
     echo 162200 > /proc/sys/cpu/0/speed
```

```
        ;;

  "resume":"standby")
        #System is resuming to Run state from Standby
        #because of activity. Increase back the CPU
        #speed.
        echo 206400 > /proc/sys/cpu/0/speed
        ;;

  "suspend":*)
        #System going to suspend state. Bring down
        #network interface.
        ifconfig eth0 down
        ;;

  "resume":"suspend")
        #System resuming from suspend state.
        #bring up network interface and
        #increase the CPU speed and
        ifconfig eth0 up
        echo 206400 > /proc/sys/cpu/0/speed
        ;;
```

Example Power State Transition

Some of the complexities involved in power state transitions can be understood by taking the example of a state transition involving both drivers and applications. Assume the system has two drivers, D1 and D2, registered with PM and three applications, A, B and C, also participating in PM (by way of opening /dev/apm_bios). Of the three applications, A and B are running with superuser privileges, and C is not. Figure 8 depicts this scenario.

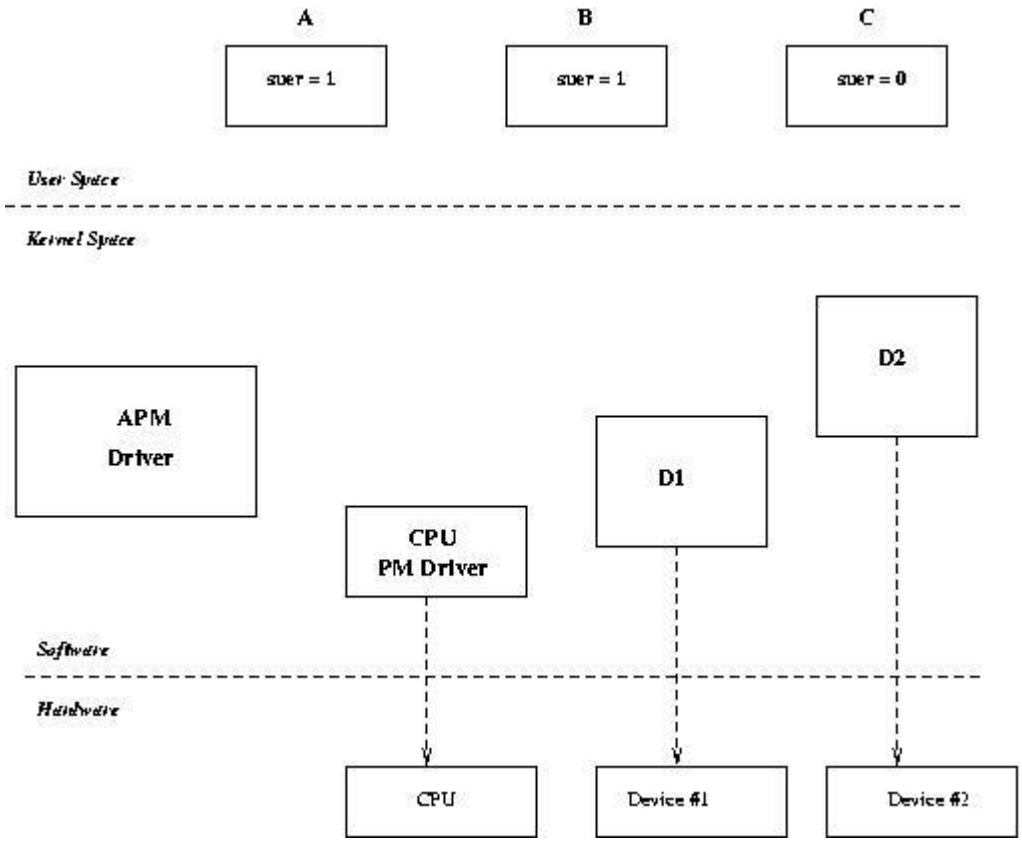


Figure 8. Example Power State Transition

Now, with this setup, let's consider a case where the system wants to transition to sleep state from run state. The sequence of steps involved in this case begins with informing applications A, B and C about the pending transition to sleep state. This allows them to take whatever actions are necessary for this transition. Also, because A and B have superuser privileges, we have to wait for them to say okay to this sleep transition before it proceeds any further.

When A and B are done with whatever work they need to perform before the system transitions to a sleep state, they give the go-ahead to the APM driver. Now, the APM driver is ready to put the system into sleep state. It sends a PM_SUSPEND message to D1 and D2. D1 and D2 put their respective devices into sleep state and say okay to APM. After D1 and D2 are finished processing this transition, APM informs the CPU PM driver to put the CPU in sleep state. At this stage, the system transition to sleep state is complete.

Conclusion

Although APM has some drawbacks, its simplicity allows it to be implemented in almost any device. Other standards, such as ACPI, provide richer control over power management at the cost of complexity. It also is essential that all device drivers and applications implement power management support correctly. Without this proper support, a single driver may prevent the system from, say, going into suspend state. Once implemented properly, power management software greatly benefits the system in terms of enhanced battery life, leading to greater efficiency.

Resources

APM Specification

Documentation/pm.txt in Kernel Sources

Intel SA1110 Advanced Developers Manual

Srivatsa Vaddagiri (vsrivatsa@in.ibm.com) has been with IBM India since 1996. He has worked on a number of projects focusing mainly on UNIX systems. Currently, he is with the embedded Linux group working on power management support for a Linux-based handheld.

Anand K. Santhanam (asanthan@in.ibm.com) has been working for IBM Global Services (Software Labs), India, since July 1999. He is a member of the Linux Group at IBM, where he concentrates primarily on device drivers, ARM-Linux and power management in embedded systems.

Vijay Sukthankar (vksuktha@in.ibm.com) has been with IBM since 1994. Currently he is managing the Linux Competency Center, and he also is managing teams working on open-source development on Linux at IBM. He also is involved in various groups within IBM to provide services on embedded Linux.

Murali Iyer (mniyer@us.ibm.com) has been with IBM since 1995 and has worked in various IBM labs around the world. Since 2000 he has been involved with designing embedded systems using Linux. Some of the projects being executed include high-end handheld devices and a programmer for pacemakers.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Bricolage Templates

Reuven M. Lerner

Issue #119, March 2004

Using a content management system doesn't mean the pages on your Web site all have to look alike. Create a custom template for each section.

Over the past few months we have looked at Bricolage, an open-source content management system (CMS) based on PostgreSQL, mod_perl and Apache. Bricolage has gained a good deal of mind share in the last few years, partly because of its open-source license, partly because of the open-source technologies on which it is based and partly because its feature set is comparable to many proprietary CMS packages, to say nothing of the open-source CMS offerings.

This month, we conclude our tour of Bricolage with a look at its use of templates. To date, we have looked at a number of the administrative and editing capabilities that Bricolage brings to the table, but a CMS is useful only if you can customize its output and make the resulting Web sites appealing.

Template Theory

HTML templates have existed for quite a long time, and they are a nice compromise between static pages and putting HTML inside of a program, which makes it inaccessible to a designer. If you want an HTML page to change dynamically, using templates is a good way to go.

Of course, this raises the important question: which templates should you use? Hundreds of different templating systems exist, including several dozen written in Perl. Some of these, such as Text::Template, are not HTML templating systems per se, but they have been used with great success on a variety of Web-related tasks.

Which templating system you should use is a debate rivaled only by Linux/BSD and Emacs/vi discussions. Luckily, Bricolage rises above the debate, coexisting

happily with both HTML::Mason and the Template Toolkit. It theoretically is possible to use another templating system, but these two are popular and powerful enough that they appear to satisfy many Bricolage users. I personally prefer HTML::Mason and demonstrate Bricolage templating with Mason, but if you are a Template Toolkit fan, feel free to use it instead.

A template is a generic outline for the content of a page of HTML. Everything is considered to be static, except for variables, whose values are filled in (interpolated, in the language of programmers) at runtime. For example, consider the following template:

```
<html>
    <head>
        <title><% $title %></title>
    </head>

    <body>
        <h1><% $headline %></h1>
        <p><% $body_text %></p>
    </body>
</html>
```

The above template, written using Mason syntax, always has a title, headline and paragraph. The contents of the title, headline and body text, however, are variable.

Mason also provides two global variables: $r, the standard mod_perl object that gives us access to the internals of our Apache server through its Perl API, and $m, the object that provides additional information about the overall Mason environment and the current, specific Mason template.

Bricolage introduces three new objects into this mix. The most important is $story, which contains information about the current story. Stories contain elements, which can themselves contain additional elements; the current element is available using the object $element. Finally, Bricolage sends pages to an output channel (normally, but not necessarily, to a Web site) using a mechanism known as a burner.

Before we continue, it's also important to understand Mason's autohandlers, which make it possible to give a site a unified look and feel. If an autohandler exists for a particular directory, then the autohandler always is going to be invoked instead of a file in that directory. That is, if you request /abc/def.html and /abc/autohandler already exists, /abc/autohandler is invoked instead of /abc/def.html. This might sound strange at first, and it is—except that the autohandler can invoke the originally requested template at any point by invoking $m->call_next().

A common strategy is to put as much common material as possible inside of the autohandler, including menus, images and headlines. The autohandler is a Mason template like any other, except for the way in which it is invoked. Inside the autohandler, between the various headlines, images and menus, you insert a call to $m->call_next(), which inserts the requested page. You thus get the benefits of a modular design using multiple templates, while simultaneously having the ability to redesign the site by changing a single file.

Autohandlers nest, meaning that Mason invokes all of the autohandlers it can find in all of its parent directories. So if we request /abc/def.html, Mason first looks for and invokes /autohandler, followed by /abc/autohandler, followed by /abc/def.html. This allows you to create a section-specific look and feel, as well as section-based menus and other information.

Modifying Templates

I'm going to assume you already have created and published at least one story on your system. If you're not sure how to create and publish a story, see my previous articles on Bricolage from the last few issues. Once you have published a story, it is sent to a particular output channel, either by copying a file on the filesystem or by using FTP to move it to a remote server.

The look and feel of your story is determined by the template. So, before we create our own simple templates, let's look at the basic examples that come with the system. Go to the Find Template link in the Template menu on the left side of the Bricolage administrator screen (user name is administrator, password is change me now!), and click on the Search button without entering anything into the text field. You should see a list of templates, one for each element type in the system (Figure 1).

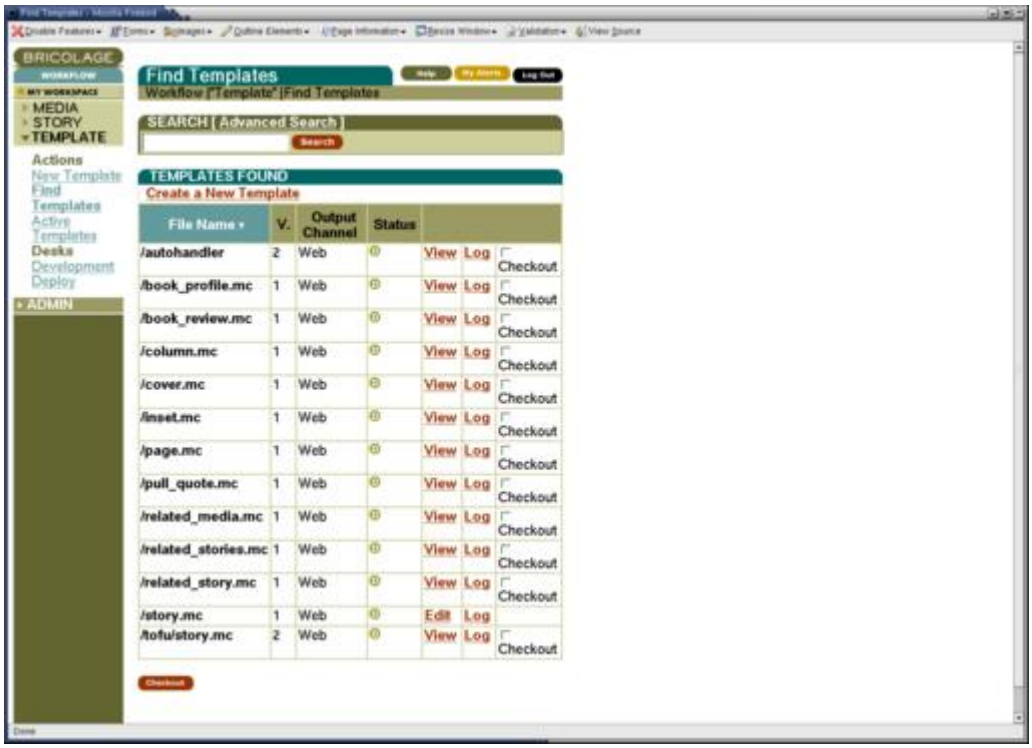


Figure 1. Select the template you want to edit from the Find Templates page.

In many ways, Bricolage templates are like stories: they are created, edited and deployed at different desks; access to them is limited to certain users and groups; and Bricolage keeps track of changes (and avoids clashes) with a simple version-control system. Indeed, if you look at Figure 1, you can see that each template has a version number associated with it. Each can be checked out of the version-control system by clicking on the associated check box and then the Checkout button at the bottom of the page. Checked-out templates are available from the Active Templates link under the Templates menu.

In Bricolage, a story is only one type of publishable element. Further, each element may contain any number of additional elements. Bricolage comes with several predefined top-level elements, such as story, book review and column, plus several additional elements designed to be included in other elements, such as a pull quote.

If you think about a daily newspaper, you should realize that each section is styled differently, even for similar elements. For example, columns in the Metro section of the *New York Times* look different from columns in the Business section, which look different from the Op-ed page. That said, they are all columns. Bricolage resolves this problem by allowing you to assign a category to an element. So if you are writing a column for the Sports section, you indicate that it is part of the sports category. When Bricolage publishes the column to the Web, it looks for the /sports/column.mc template. If it exists, Bricolage applies that specific template. If not, Bricolage looks for a column.mc template at the top (root) category. In other words, if you have a top-level

template for an element, it serves as a fallback, or default, for all the elements of that type on the system. You can give some or all sections of your site a different look and feel by defining category-specific templates.

As you can see from Figure 2, I have checked out the /story.mc template, which is the top-level template for stories. Rather than having a view link, I have an edit link that allows me to modify the template. I also can edit the template by going to the active templates page, which provides me with a similar edit link. Open up the template for editing, which should give you a screen similar to Figure 2.

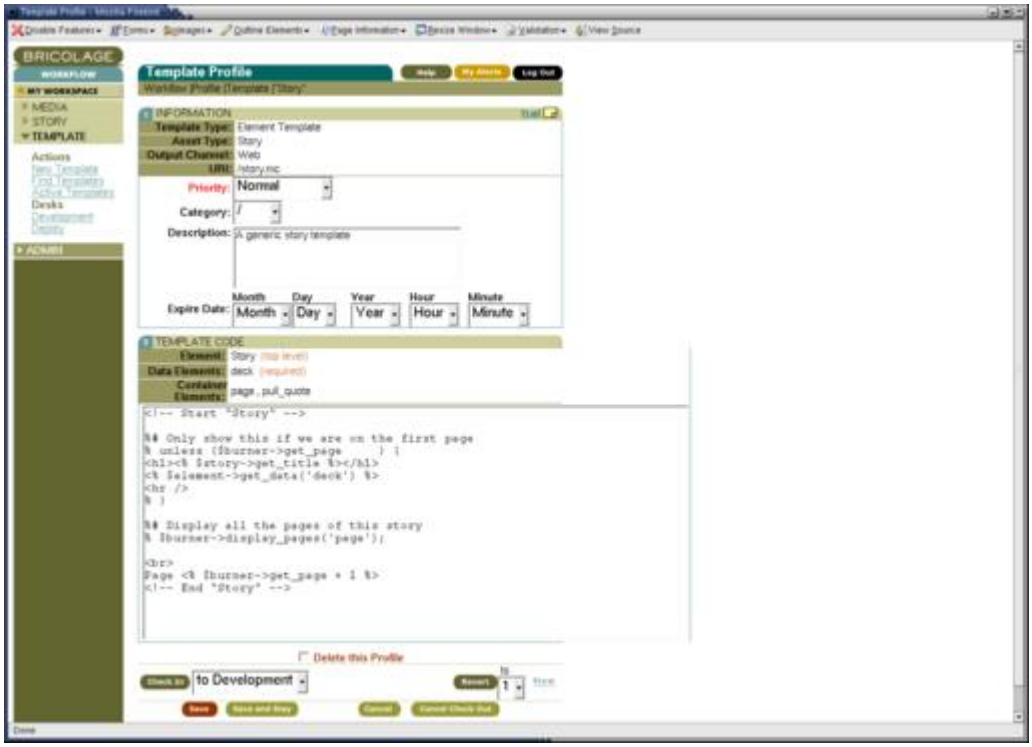


Figure 2. Editing a Checked-Out Template

Editing a template is similar to editing a story or any other element type, except that you are modifying the container into which the story will be inserted. If you insert only static HTML, every element looks identical. Thus, the trick is to use the predefined $story, $element and $burner objects to fill in the page with dynamic content. For example, here is the default /story.mc template:

```
<!-- Start "Story" -->

%# Only show this if we are on the first page
% unless ($burner->get_page     ) {
<h1><% $story->get_title %></h1>
<% $element->get_data('deck') %>
<hr />
% }

%# Display all the pages of this story
% $burner->display_pages('page');

<br>
Page <% $burner->get_page + 1 %>
```

```
<!-- End "Story" -->
```

As you can see, the above template is rather simple. An actual site might insert a drop-cap, set some styles in CSS or include some additional static text to identify the site. The basic version of /story.mc does the following:

- It gets the current page number from $burner->get_page(). The page numbering begins at 0; however, we display the story's title and the element's deck if we are on the first page. The title comes from the $story object, with $story->get_title(), and the deck (an abstract) comes from the element itself. Notice how $element->get_data() is a fairly generic-looking method; we can use $element->it to retrieve any field from within an element.
- We display the story by requesting it from the burner, using $burner->display_pages('page').
- Finally, we use $burner->get_page() once again to show the page number at the bottom of the page.

What happens if we remove the page number and insert some static HTML of our own at the top or bottom of this template? Our changes then are reflected for all stories on the system. But it's important to remember that not all elements are stories, so changes that we make to /story.mc do not affect columns, book reviews or other element types.

When you are done editing /story.mc, you can click on the Check-In button at the bottom of the page. When you check a template in, you can send it to the development template desk (the default) or you can deploy it right away. This option is particularly useful if you discover a mistake on a template that is affecting the entire site; you can modify the template, deploy it and see the results immediately.

Finally, notice how /story.mc does not contain any <html> or <title> tags. That's because such items are implemented within the autohandler. The /autohandler template, which you can view, check out and edit from the Templates menu, is defined by default to be the following:

```
<!-- Start "autohandler" -->
<html>
    <head>
        <title><% $story->get_title %></title>
    </head>
    <body>

    % $burner->chain_next;
    </body>
</html>
<!-- End "autohandler" -->
```

The autohandler, which is global to the entire site, places the title of a story within the appropriate HTML <title> tags. It also incorporates the appropriate page contents into the template by invoking $burner->chain_next().

If you want to include a global CSS stylesheet, add a standard menu to the top of each page or put your company's logo at the top of each page on the site, this autohandler is the place to do it. And because autohandlers nest, you can have a global autohandler for your overall site, with section-specific autohandlers in each category.

Creating Templates

So far, we have looked only at existing templates. It's quite easy to create a new template, though. Simply go to the Template menu and click on New Template. You should see a screen that looks like the one shown in Figure 3, asking you to indicate the output channel and category to which your template should apply. Click on Next, and you are asked for the element type to which your template should apply.

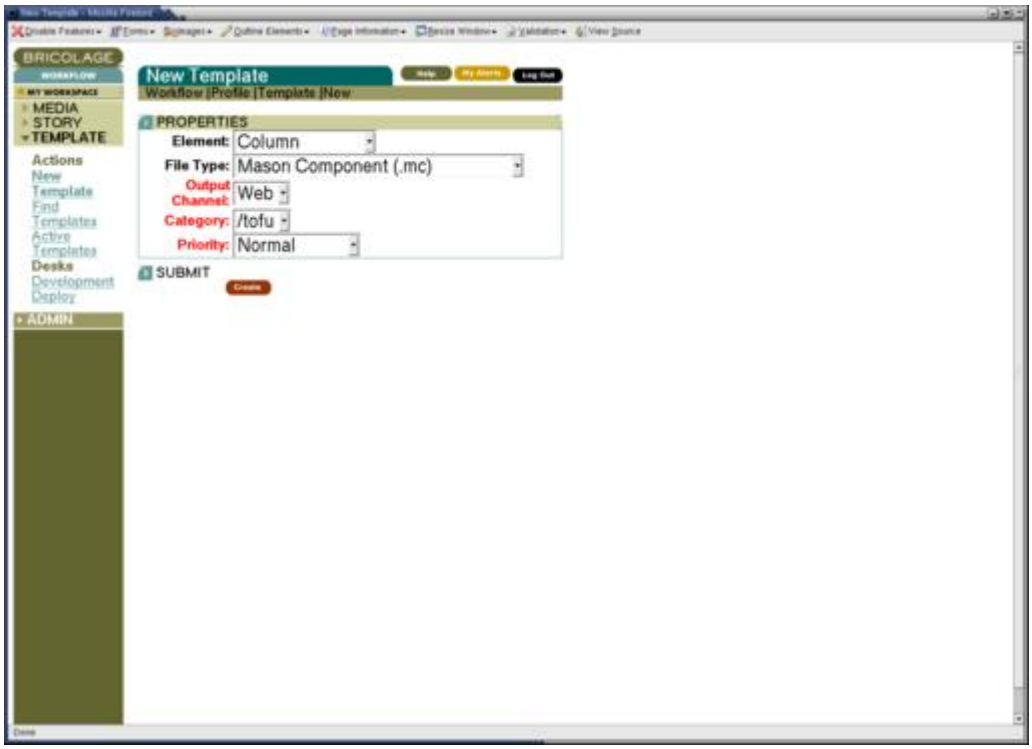


Figure 3. Selecting the Output Channel and Category for a New Template

The category-channel-element combination must be unique. You therefore can have multiple templates for an output channel, for a category or for a particular element. But for story elements in the Web output channel in the root (/) category, there can be only one template. If you try to violate this uniqueness constraint, Bricolage issues a warning, telling you that there already is a template for that combination. There are several solutions to this problem; one is to create a new element type, another is to create a new category and still

another is to modify the existing template for that combination. The best course of action depends on your specific goals.

I'm going to create a new template for columns within the tofu category, which is represented as /tofu/column.mc in Bricolage. Once I click the Create button, I'm presented with an editing screen that allows me to create or modify my template. I'm going to make my template extremely simple:

```
<!-- Start "tofu/column" -->

%# Display this story
% $burner->display_pages('page');

<!-- End "tofu/column" -->
```

Notice how we put HTML comments around the definition. This makes it easier to debug the template when it is turned into HTML and sent to the user's browser. I can assure you from personal experience that the nested nature of Mason templates, especially with multiple autohandlers, can be maddening.

Once I select and deploy from the Check-In menu, and then click the Check-In button, my template is deployed. Any column with a category of tofu now is formatted with this template rather than the global, more general one for columns.

And, of course, if I want to go back and edit my template, I can do that in the way that we saw earlier—finding it, checking it out and editing it.

Conclusion

Bricolage, like any serious CMS, makes it easy to create a unified look and feel by using templates. Because Bricolage is based on standard open-source tools, such as mod_perl and Apache, it can take advantage of the existing templating systems for mod_perl, including HTML::Mason and the Template Toolkit. This month, we saw how we can create and modify templates associated with various element types and categories, giving us the flexibility we need to generalize a site's look and feel without being constrained.

Resources

The main source of information about Bricolage is the project's Web site, bricolage.cc. This site has pointers to downloadable source code (hosted at SourceForge), documentation and an instance of Bugzilla (bugzilla.bricolage.cc) for bug reports and feature requests.

SourceForge hosts several Bricolage mailing lists, in which the developers participate actively. If you have questions or want to learn about new releases, you can subscribe from the SourceForge page, sourceforge.net/projects/bricolage.

The Bricolage documentation generally is quite good, if technical. A more user-level introduction to the system was published by O'Reilly and Associates as an appendix to their recently published book about Mason. You can read that appendix on-line at www.masonbook.com/book/appendix-d.mhtml.

You can learn more about Mason from both the Mason book site, www.masonbook.com, and the Mason home page, www.masonhq.com.

Finally, you can learn more about David Wheeler (the primary author and maintainer of Bricolage) at david.wheeler.net, and about his company, Kineticode, at www.kineticode.com.

Reuven M. Lerner, a longtime consultant in Web/database programming, is now a graduate student in Learning Sciences at Northwestern University in Evanston, Illinois. You can reach him at reuven@lerner.co.il.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Kernel Korner

What's New in the 2.6 Scheduler?

Rick Lindsley

Issue #119, March 2004

When large SMP systems started spending more time scheduling processes than running them, it was time for a change. The new Linux scheduler works well on systems with many processors.

As work began on the 2.5 Linux kernel tree back in December 2001, there was a lot of talk in the community about scaling. Linux had begun to appear in some of the roles traditionally filled by larger servers, and several vendors were offering versions of Linux suitable for symmetric multi-processing (SMP). Although commercial interest in that area seemed to be growing, there also was a growing realization that even SMP Linux wasn't scaling as well as it should. If, say, two single-processor desktop machines could outperform a single four-processor machine, who'd want to use (or buy) the four-way?

One of the first areas of the kernel that required attention was the scheduler. It became apparent that as the load and the number of CPUs increased, the scheduler worked harder and harder and ended up taking more and more time away from the processes it was scheduling. In the worst case, nearly the entire system was consumed trying to decide what to run next.

When we speak of the Linux scheduler, we're not referring to a specific task that handles all the scheduling. Rather, each task itself does a little bit of the scheduling each time it acquires or yields the processor, by calling a scheduler function within the kernel. So when we speak about the scheduler doing this or that, we really mean the scheduler function and its related routines in the context of some other task.

The 2.4 Scheduler

The original 2.4 scheduler was quite simple. All tasks on the system were already on a global list called tasklist, and these were assigned a goodness rating. Goodness was determined by:

- How many clock ticks you might have left: when a task is given the processor, the task is allocated a certain amount of time to use it before that task is interrupted involuntarily and replaced by another task. If it gives up the processor voluntarily—to wait for I/O, for example—then the task's generosity would be rewarded by being at a higher priority to regain the processor once the I/O job was complete.
- CPU affinity: by using another system call, it is possible to advise the scheduler that you wish to remain on a particular processor, even if another processor should free up first.
- Nice or user-set priority: if the user is root, it's possible for the user to increase or decrease the priority of a task within a fairly substantial range.
- Whether the task was a real-time task: a task that has been designated a real-time task has a higher priority than all tasks that are not real time.

So when a processor came free, the 2.4 scheduler would examine the tasklist, looking for the task with the highest goodness, and select that task for running next. Figure 1 demonstrates how the 2.4 scheduler worked. The tasklist *was* the runqueue, and because it wasn't ordered in any helpful way, each iteration of the scheduler would examine the tasklist completely, looking for the best candidate for the idle processor. In the case of multiple processors, it was a matter of chance if you ended up on the same processor twice in a row, even if you were the only runnable task.

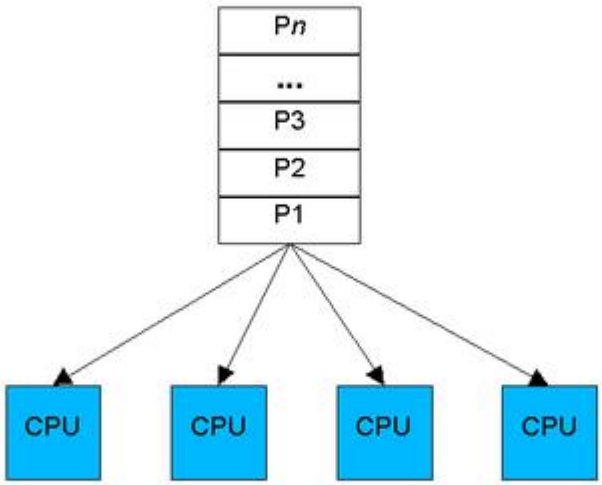


Figure 1. The 2.4 scheduler was shared among processors and wasn't ordered in any helpful way.

This model had the advantage of being quite simple to implement and fairly simple to debug. The tasklist was guarded by a single read/write spinlock. This allowed multiple tasks to examine it in parallel while still providing the mechanism for obtaining exclusive access for the comparatively rare event of changing it.

Unfortunately, these same features also were the model's disadvantages. Instrumentation of the then-current 2.4 scheduler began to zero-in on the problem: the single read/write spinlock tended to become a point of contention on both busy systems and systems with four or more CPUs. Only a single queue was used for all processors, and it had to be examined completely for each reschedule. As the system got busier, the tasklist got longer; the linear search for the best task took longer as well. As a result, having decided which process to run, you waited longer to acquire the lock exclusively to remove that task from the runnable list and mark it running. If the wait was long enough, several processors might choose the same process only to learn that it already had been given to a different processor. The other processors then would have to go back to the linear search and find another task. As the system got busier, the scheduler consumed more CPU time, to the point where scheduling processes took more time than did running them. Changes needed to be made so that a loaded system didn't schedule itself into a standstill.

The 2.6 Scheduler

Thus, the stage was set for the introduction of the O(1) scheduler in 2.6, which boasts that the time to select the best task and get it on a processor is constant, regardless of the load on the system or the number of CPUs for which it is scheduling. Instead of one queue for the whole system, one active queue is created for each of the 140 possible priorities for each CPU. As tasks gain or lose priority, they are dropped into the appropriate queue on the processor on which they'd last run. It is now a trivial matter to find the highest priority task for a particular processor. A bitmap indicates which queues are not empty, and the individual queues are FIFO lists. Therefore, you can execute an efficient find-first-bit instruction over a set of 32-bit bitmaps and then take the first task off the indicated queue every time.

As tasks complete their timeslices, they go into a set of 140 parallel queues per processor, named the expired queues. When the active queue is empty, a simple pointer assignment can cause the expired queue to become the active queue again, making turnaround quite efficient.

It's not possible to draw the 140 queues now present for each CPU without resorting to mere dots. But, Figure 2 offers an approximation and drives home the major difference between the 2.4 and the 2.6 schedulers. Except on a heavily loaded system, most queues are empty. Those that are not empty have their best selection at the head of the queue, so searching for the next task to run has become easy.

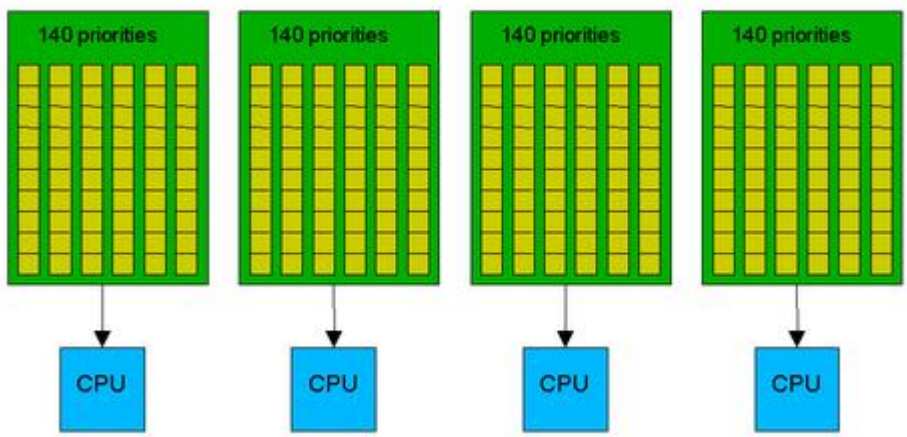


Figure 2. The 2.6 scheduler has 140 queues per processor, making it easy to search for the next runnable task.

There's one shortcoming of this 2.6 method. Once a task lands on a processor, it might use up its timeslice and get put back on a prioritized queue for rerunning—but how might it ever end up on another processor? In fact, if all the tasks on one processor exit, might not one processor stand idle while another round-robins three, ten or several dozen other tasks? To address this basic issue, the 2.6 scheduler must, on occasion, see if cross-CPU balancing is needed. It also is a requirement now because, as mentioned previously, it's possible for one CPU to be busy while another sits idle. Waiting to balance queues until tasks are about to complete their timeslices tends to leave CPUs idle too long. Instead, 2.5 and 2.6 leverage the process accounting, which is driven from clock ticks, to inspect the queues regularly. Every 200ms a processor checks to see if any other processor is out of balance and needs to be balanced with that processor. If the processor is idle, it checks every 1ms so as to get started on a real task as soon as possible.

This is not to say that the scheduler is fixed now and all work on it has stopped. Some workloads and architectures provide some interesting scenarios that the scheduler still doesn't deal with well.

Current and Future Work

The goals of a successful scheduler can be stated simply, even if they always can't be attained simply.

1. Minimize the time spent scheduling, so as to maximize the time spent executing.
2. On multiple CPUs, keep the load spread around so it is easier to share the processors fairly.
3. Provide good response to interactive programs.

In addition, the philosophy of the Linux scheduler is that it should be mostly right all of the time rather than perfect much of the time. Even though different workloads exhibit different behaviors and place different stresses on the system, the scheduler should be sufficiently general and robust so that all workloads are handled at least adequately, without additional tuning being necessary.

Interactivity

Most of the scheduler tweaking in 2.6 has been done in an attempt to improve interactive response. Originally, this was interactive in the traditional sense of dragging windows across a monitor or typing on a keyboard. An interactive task was meant to define a task that utilized a lot of human interaction. But it gradually has been expanded to mean “tasks that should receive high priority upon waking up from self-imposed sleeps”. This includes the previous set of

tasks but also now includes tasks for which a delay is noticeable by humans, such as delays in music players. Because this is a subjective evaluation, it might never be resolved to everyone's satisfaction. General agreement from testers, however, is the situation is better now with the 2.6 scheduler.

Process Affinity

Imagine two tasks spending a lot of time communicating with each other over a pipe or bit of shared memory. Some evidence exists that they might do better if they were on the *same* processor, even if that means leaving another idle. If one sends a message to the other and then waits for a reply, both tend to alternate being runnable with small overlaps where they are both runnable. In other words, they don't collide often. The big savings comes from having the processor cache pre-warmed with the data in the pipe so it doesn't need to be fetched and copied again to a different processor's cache. Although processor speeds have been increased, memory and bus speeds have not kept pace. It's becoming increasingly painful to have to retrieve data that used to be cached.

Process Size

Moving a task that uses little memory affects a processor differently from moving a task that uses a great deal of memory. However, either the large or small task may be the correct choice depending on the circumstances. If you move a task that uses a lot of memory away from a processor, leaving behind many small tasks that don't use much memory, each of those tasks may find a higher percentage of their memory in cache each time they run. On the other hand, moving that large task to another processor that has large tasks may now cause *all* the tasks to start with a cold cache and negatively affect the performance of both it and its new neighbors. Current code does not take process size into account at all.

Device Affinity

For much the same reason as process affinity, there might be times when it pays to overload a processor with tasks if those tasks are making heavy use of a particular device. Web servers, for instance, often have multiple network cards but not enough to have one for each CPU. Congregating those tasks on processors where the network data is arriving might prove quite advantageous. Determining which tasks are likely to use which devices is currently neither obvious nor easy.

Heavy Spikes but Short-Lived Tasks

Shell scripts can cause an explosive number of short-lived tasks, especially if they don't or can't use built-in functions for string or number manipulations.

Although one could argue these are poorly coded scripts, they nevertheless have demonstrated that the scheduler can be too slow in balancing queues on SMP machines. Workloads with similar behavior also would suffer.

Light but Unbalanced Load

Imagine a program that divides the work into six equal pieces, all of which ideally finish at the same time. Unfortunately, on a four-processor machine, two processors tend to take two tasks and two tend to take one task, and things stay that way. Unless the scheduler makes an effort to spread the pain around, two jobs finish more quickly than the other four because they have no competition on their processor. On the other hand, in most job mixes, moving those odd jobs around still leaves two tasks on each of two processors.

NUMA

NUMA (non-uniform memory access) presents some interesting characteristics to worry about. In a NUMA system, it may be much more expensive to get memory from over there, near another processor, than from here, near this processor. It's not sufficient to have an idle processor; you need one that is both idle and not too expensive to migrate to. For instance, it might be bad to migrate a task if most of the task's resources reside at the current location. It even might be so bad that it's better to leave it on a busy processor than to move it to an idle one with a cold cache.

Hyperthreading

Hyperthreading introduces yet another complexity. In hyperthreading, two processors share cores that overlap in a hardware-dependent way. Because of this interdependency, jobs running on one processor *can* affect the speed of a job running on the other processor. Although nobody would ever expect a box with four hyperthreading processors to equal a full eight-processor machine, exactly what to expect varies a great deal by workload. The only sure thing is it should not yield *less* performance.

Summary

The 2.6 scheduler offers some strong improvements over the 2.4 scheduler. It scales better on larger machines for most workloads without giving up the performance demanded by the one- and two-processor users that make up much of the Linux market. Recent changes allow the scheduler to handle kernel builds smoothly at the same time as it plays your favorite songs. The 2.6 kernel is available now to adventurous souls at www.kernel.org. Full distributions from Linux vendors that utilize 2.6 kernels will lag as vendors complete their own

testing and add their own support features; you should contact your favorite vendor directly for release information.

Rick Lindsley has worked with UNIX and Linux for 20 years. He's currently working on Linux scheduler improvements in IBM's Linux Technology Center and can be reached at ricklind@us.ibm.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Cooking with Linux

Can't Get Enough Desktops!

Marcel Gagné

Issue #119, March 2004

Explore new user interface ideas without leaving your regular desktop. Our chef shows you how to nest X sessions.

Ah, François, I see you have decided to run Window Maker—excellent. It's a great window manager, and I think you'll enjoy it. Two weeks ago, you were running GNOME; last week it was KDE, and now it's Window Maker. I'm glad you took my advice and decided to try other desktop window managers. Choice, after all, is one of the great joys of running Linux.

Quoi? Which one am I running? All of those and a half-dozen more, *mon ami*. François, I am not pulling your leg. My main desktop is KDE, but over here I have GNOME and over there XFCE, and in that virtual desktop, I have Window Maker. Furthermore, I have IceWM running in my Window Maker session. No, *mon ami*, it is not complicated at all, and I will show you how it is done as soon as our guests arrive. But they are already here! François! *Vite*! To the wine cellar. Given that today's menu is more dessert than meal, fetch the 2001 Niagara Peninsula Riesling ice wine and bring it back *tout de suite*!

Welcome, *mes amis*, to *Chez Marcel*. Please sit and make yourselves comfortable. I have been pushing my faithful waiter to experiment with different desktops. It's always nice to try something different, which is why we change the menu from time to time, *non*? The same is true for your desktop environment. KDE or GNOME may be your favorite, but why not try Window Maker, IceWM or XFCE for a change? A little visit to Matt Chapman's Window Managers for X Web site at www.plig.org/xwinman should whet your alternative window manager appetite. In fact, why not try them while you are still running your favorite desktop? *Non, mes amis*, I have not been

oversampling the wine. It is possible to run multiple desktops simultaneously, and it is a lot of fun once you get the hang of it.

As with many things in the Open Source world, there is certainly more than one way to do it. The first involves jumping out of your current X session back to one of your virtual terminals. If you already are running an X session, press Ctrl-Alt-F1 and you should find yourself back at a text screen. Incidentally, F1 just as easily could be F2, F3, F4, F5 or F6. If you started X from the command line (as opposed to using a login manager like gdm or kdm), you should see the dialog for the session when you press Ctrl-Alt-F1 with the log output from X on your screen. Any of the other function keys, F2–F6, should provide you with a text-based login screen. Simply press Ctrl-Alt-F2 for virtual terminal two and so on.

Your graphical session is still active. It runs by default on what the system calls display :0, something you can verify by typing the command `echo $DISPLAY` at a shell prompt (inside your graphical session, of course). You should see your PC's hostname with the display suffix. Press Ctrl-Alt-F7 to go back to your KDE, GNOME or whatever session. Go ahead and try it, and then head on back to a text screen (Ctrl-Alt-F?). Your X session, then, is on virtual terminal seven. From the text screen, log in as yourself and type the following:

```
xinit /usr/X11R6/bin/xterm -- :1
```

Notice the `-- :1` at the end of this line. Because X already is running on display :0, we need to run this X terminal on an alternate display, in this case, :1. Now a new X session begins, this one running on virtual terminal eight. It looks pretty boring because all you have is a simple X terminal running on a gray background. There isn't much to look at or even any way to move the X terminal window around, but you can execute commands and even start up other X programs, which you also won't be able to move around. To go from your new session to the old, press Ctrl-Alt-F7, then Ctrl-Alt-F8 to get back to your X terminal. Easy, *non*? Using this technique, you could start something more interesting, like another window manager, such as IceWM or XFCE, and happily switch back and forth from one virtual X session to another.

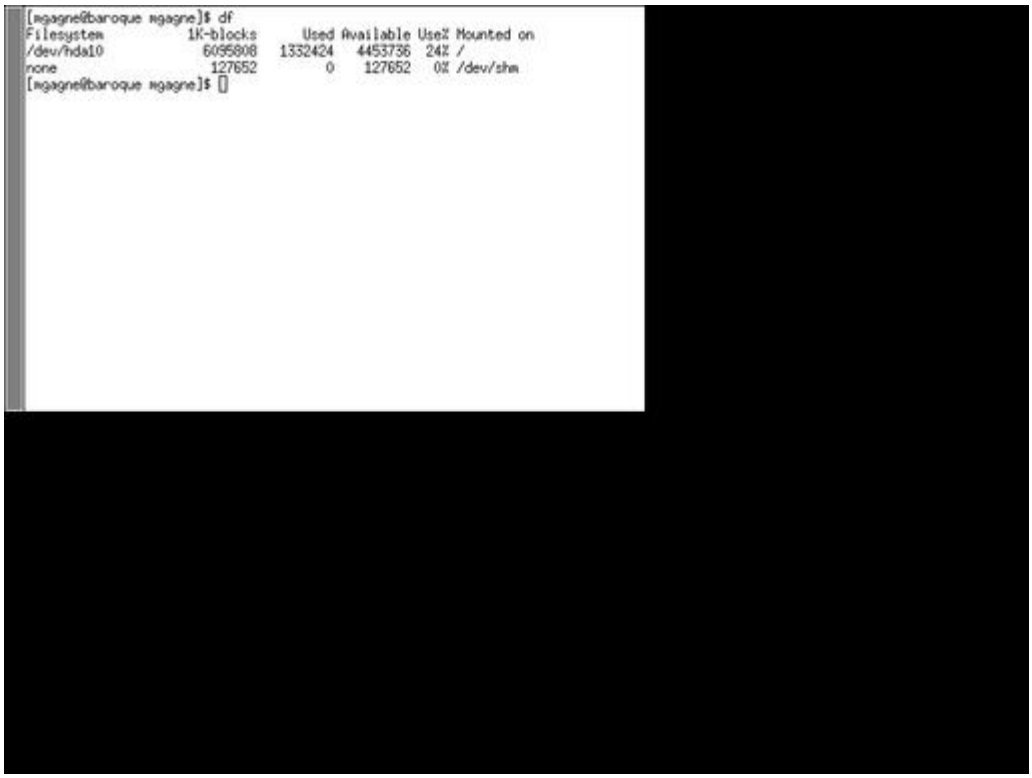


Figure 1. A Somewhat Dull and Bare X Display with a Single X Terminal

To get out of that X session, you should know about the classic X window escape hatch, what I call the "*oh, mon Dieu*, I've tried everything and I can't get out of X" escape clause. Simply press Ctrl-Alt-Backspace. This is, *mes amis*, a rather rude way to exit X and should be used only when no other options are available.

Switching from one X session to another is fine, but doing this makes it hard to continue working on two desktops. To avoid constantly switching back and forth, I have been running a program called **Xnest**. Xnest, part of the XFree86 distribution, is interesting because it is both an X client and server all in one. It literally is a nested X server. Before I continue, I probably should let you know that although it is part of X, it may not be installed already on your system; however, the package (XFree86-Xnest) is likely on your distribution CDs.

In order to start a nested server, you have to provide an alternative DISPLAY variable, as we did with the X terminal earlier. Because your own X server is probably running as :0, choose :1. To make sure you can connect to this new server from any of your applications, use the -ac option as well. This option disables access controls. The ampersand starts my new server as a background process:

```
Xnest :1 -ac &
```

Now, a blank window starts on your desktop with the basic X cursor in the center. On my Mandrake system, it was a dark-blue square. On another server running Red Hat, it was black. It doesn't look like much, so let's start an X application on our new server. We start with something simple like our venerable Xclock:

```
xclock -display :1
```

When you press Enter, the classic Xclock appears in your second X server window. This also is a great way to play with things like X resources. For instance, let's dress up that rather boring clock and move it to another part of the display:

```
xclock -foreground "Red" \
-geometry +450+250 -display :1
```

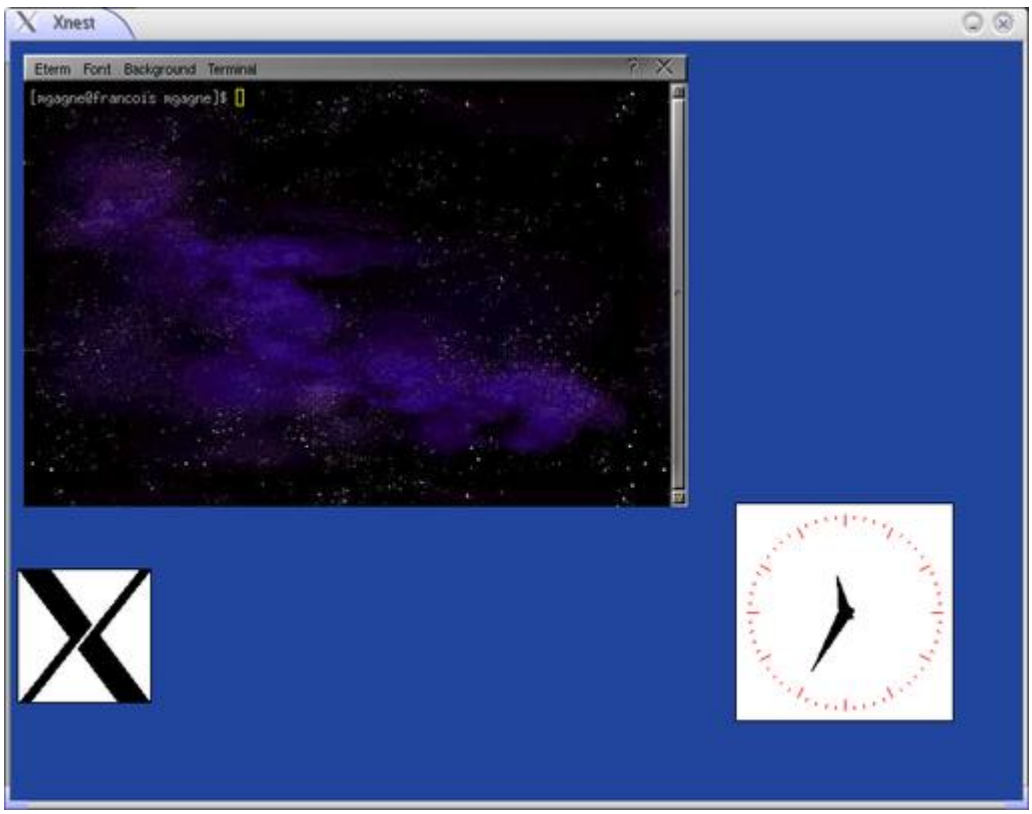


Figure 2. Populating Your Nested X Server

One by one, you could populate this new server with applications—the X logo here and an Eterm there. Of course, moving windows around isn't possible in this environment, thereby making this idea only so useful. To experience the flavor of desktop mania truly, you need the whole smörgåsbord, meaning a window manager and, of course, a little more wine to go with it.

Starting a full-blown window manager is a similar process, and for this next step we start with the basic Tabbed Window Manager (TWM). This is the most basic window manager you have and comes as part of XFree86. Begin by

closing the applications in your Xnest so you can start clean. You should be looking at that blank square with the X cursor in the center. Now, from the command line, type:

```
twm -display :1
```

If nothing changes, press the left mouse button and TWM's menu should appear. I did say it was a basic window manager, *non*? Let's try running Window Maker this time:

```
wmaker -display :1
```

As you can see, the format essentially is the same, passing the -display parameter to the window manager's command name in each case. This is, of course, where I should tell you that not every window manager uses the same switch. Here's a list of the more popular window managers and what you need to start them:

- Motif Window Manager: `mwm -display :1`
- F Virtual Window Manager: `fvwm2 -display :1`
- GNOME (note double hyphen): `gnome-session --display :1`
- AfterStep: `afterstep -d :1`

In some cases, you won't be able to start a window manager using some kind of display redirect switch. This is true with KDE, AmiWM, XFCE and some others. To run these window managers, begin with a simple X terminal in your nested X server:

```
xterm -display :1
```

From that command line, start your window manager or desktop simply by typing its command name, such as `xfce4-session` for XFCE, `amiwm` for AmiWM and so on.

After you've played with several window managers in this manner, you may find yourself faced with a bit of a roadblock. Sometimes, in doing this, I ran into a message (particularly from GNOME) that it could not start, specifically `gnome-session: you're already running a session manager`. As I knew this wasn't the case, I checked to see what the SESSION_MANAGER pointed to:

```
$ echo $SESSION_MANAGER
local/ultraman:/tmp/.ICE-unix/3132
```

As you can see, I had leftover session information from a prior run with a nested window manager. One option is to unset the SESSION_MANAGER variable. Another is simply to remove the troubling files, assuming, of course, that you are no longer running your window manager:

```
$ rm /tmp/.ICE-unix/3132
rm: remove socket `/tmp/.ICE-unix/3132'? y
```

It would appear, *mes amis*, that closing time is approaching rapidly, so it's time for *la pièce de résistance*—your desktop dessert, if you will. Some of you may be asking, "if I can run one nested server, why not two or three?" Starting a second nested server is simply a matter of assigning a different display number. For the second, type `Xnest -ac :2` or `Xnest -ac :3` for the third and so on. In fact, you can run an Xnest inside of another Xnest. Figure 3 shows an IceWM session running inside a GNOME session, running inside a KDE session.

Figure 3. IceWM Nested inside of GNOME, Nested inside of KDE

As you can see, *mes amis*, there is enough here to satisfy the greatest desktop gourmand among you, *non*? As Robert Heinlein might have said, "to enjoy the full flavor of life, take big bites" or in this case, run many different window managers and many desktops.

And now, closing time truly is upon us, but spend a little more time exploring. François will refill your glasses one final time before you go. Until next time, *mes amis*, let us all drink to one another's health. *A vôtre santé! Bon appétit!*

Resources

Window Managers for X: www.plig.org/xwinman

XFree86 Web Site: www.xfree86.org

Marcel's Web Site (check out the wine page): www.marcelgagne.com

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of the newly published *Moving to Linux: Kiss the Blue Screen of Death Goodbye!* (ISBN 0-321-15998-5) from Addison Wesley. His first book is the highly acclaimed *Linux System Administration: A User's Guide* (ISBN 0-201-71934-7). In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Paranoid Penguin

Application Proxying with Zorp, Part I

Mick Bauer

Issue #119, March 2004

An application-level proxy blocks the widest possible range of network attacks but is more complex than a packet filter. Is the trade-off worth it?

At first glance, stateful packet filtering appears to have conquered the firewall world, both in terms of market share and mind share. The list of products based on stateful packet filtering is a long one, and it includes both the proprietary industry leader, Check Point Firewall-1, and Linux's excellent Netfilter kernel code.

But what about application-layer proxies? Professional firewall engineers have long insisted there's nothing like an application-aware proxy for blocking the widest possible range of network attacks. Indeed, being such a person myself, I've been disheartened to see application-layer proxies increasingly marginalized. In some circles they've even been written off as obsolete for reasons that simply don't warrant, in my opinion, the loss of a powerful security tool. Marketing is at least as big a reason as any other.

Apparently I'm not alone in my opinion. Balazs Scheidler, creator of the essential logging facility Syslog-NG, has created Zorp, an open-source proxy firewall product that is simply brilliant. This month I explain why Zorp has helped resuscitate my faith in the application-layer proxy firewall, and what this means for anyone charged with protecting highly sensitive networks.

Firewall Refresher Course

At this point, some of you may be asking, “What are application-layer proxying and stateful inspection? And why do I care which is better?” I can explain. Feel free to skip ahead to the next section if you're a grizzled firewall veteran.

A firewall, of course, is a computer or embedded hardware device that separates different networks from one another and regulates what traffic may pass between them. The instructions that determine which network nodes may send what type of network packets and where are called firewall rules or, collectively, the firewall policy.

These rules are what make a firewall different from an ordinary router. Routers must be programmed to know how to move packets from one network to another, but not necessarily whether to allow them to move in any given way. A firewall, on the other hand, discriminates.

One very simple way to categorize packets is by the Internet information in packets' Internet Protocol (IP) headers. An IP header contains basic information, most importantly, protocol type, source and destination addresses, and, if applicable, source and destination ports. The ports actually are part of the next header down in the packet, the UDP header or TCP header. A firewall that looks only at this basic information is called a simple packet filter. Because simple packet filters don't look deeply into each packet, they tend to be quite fast.

However, the IP header of a packet plus its TCP or UDP port number tells us nothing about that packet's relationship to other packets. For example, if we examine the IP header of an HTTP packet, we know it's a TCP packet (thanks to the IP field), where it's from and where it's going (source and destination IP address fields) and what type of application sent it (from the destination port, TCP 80). Table 1 shows an example simple packet-filtering rule.

Table 1. Simple Packet Filter Rules for HTTP

Source IP	Destination IP	Protocol	Source Port	Destination Port	Action
Any	192.168.1.1	TCP	Any	80	Allow
192.168.1.1	Any	TCP	80	Any	Allow

But that level of inspection leaves out some key pieces of information about the HTTP connection: whether the packet is establishing a new HTTP session, whether it's part of a session in progress or whether it's simply a random, possibly hostile, packet not correlating to anything at all. This information is left out because crucial session-related information such as TCP flags, TCP sequence numbers and application-level commands, all are contained deeper within the packet than a packet filter digs. That's where stateful packet filtering comes in.

A stateful packet filter, like a simple packet filter, begins by examining each packet's source and destination IP addresses, and source and destination ports. But it also digs deeper into the packet's UDP or TCP header to determine whether the packet is initiating a new connection. If it is, the firewall creates an entry for the new connection in a state table. If it isn't, the stateful packet filter checks the packet against the state table to see if it belongs to an existing connection. A stateful packet filter will block packets that pretend to be part of an existing connection, but aren't. Actually, UDP is connectionless, but a good stateful firewall can guess that an outbound DNS query to a given server on UDP 53 should be followed by an inbound response from that server's UDP port 53. Stateful packet filtering has two main benefits over simple packet filtering.

First, firewall rules can be simpler. Rather than needing to describe both directions of each bi-directional transaction, such as HTTP, firewall rules need address only the initiation of each allowed transaction. Subsequent packets belonging to established, allowed connections can be handled by the firewall's state table, independently of explicit rules. In Table 2 we see that only one rule is needed to allow the same HTTP transaction for which we needed two rules in Table 1.

Table 2. Stateful Packet Filter Rule for HTTP

Source IP	Destination IP	Protocol	Source Port	Destination Port	State	Action
Any	192.168.1.1	TCP	Any	80	New	Allow

The second main benefit of stateful packet filtering is we don't have to do such distasteful things as allowing all inbound TCP and UDP packets from the Internet to enter our internal network if they have a destination port higher than 1024. This is the sort of thing you sometimes must do if you don't have a better way to correlate packets with allowed transactions. In other words, stateful packet filtering provides better security than simple packet filtering.

“Cool”, you say, “stateful packet filters are more efficient and secure”, which is true. But what about the things even stateful packet filters don't consider? What about things like potentially malformed HTTP commands or intentionally overlapping IP fragments? Might there be a type of firewall that examines each packet in its entirety or that has some other means of propagating the fewest anomalous packets possible?

Indeed there is, and it's called an application-layer proxy or application-layer gateway. Whereas packet filters, whether simple or stateful, examine all packets and pass those that are allowed, an application-layer proxy breaks

each attempted connection into two, inserting itself in the middle of each transaction as an equal participant. To the client or initiator in each transaction, the firewall acts as the server. To the intended destination, or server, the firewall acts as the client.

Figures 1 and 2 illustrate this difference. In Figure 1, we see that the stateful packet filter passes or blocks transactions but ultimately is an observer in that it passes allowed packets more or less intact, unless, for example, it performs network address translation (NAT). In contrast, in Figure 2 we see that the firewall terminates each allowed connection to itself and initiates a new, proxied connection to each allowed connection's desired actual endpoint.

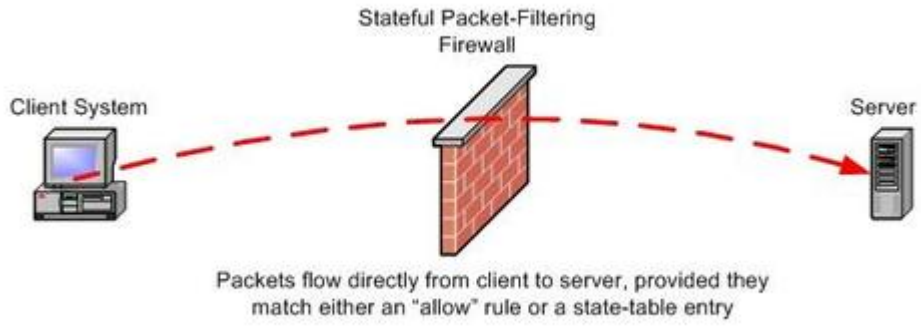


Figure 1. With a stateful packet filter, packets flow directly from client to server, provided they match either an allow rule or a state-table entry.

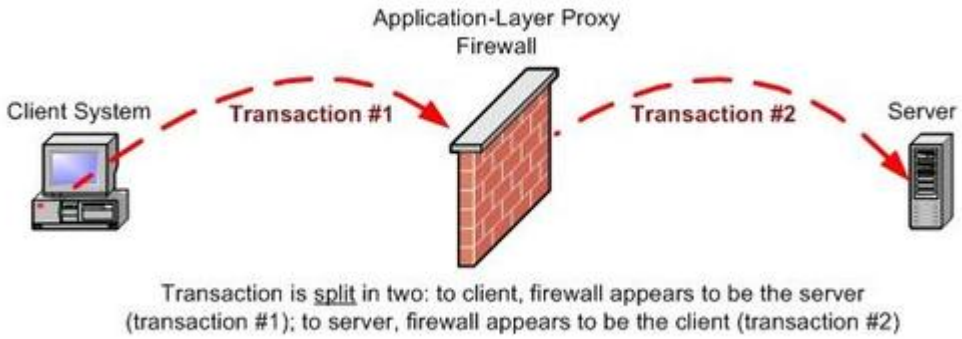


Figure 2. With an application-layer proxy, the connection is split in two. To the client, the firewall appears to be the server (transaction #1). To the server, the firewall appears to be the client (transaction #2).

Proxying comes in two flavors, transparent and nontransparent. In a transparently proxied connection, both parties are unaware that the connection is being proxied; the client system addresses its packets as though there were no firewall, with their true destination IP address. By contrast, in a nontransparently proxied connection the client must address its packets to the firewall rather than to their true destinations. Because the client must, in that case, somehow tell the firewall where to proxy the connection, nontransparent

proxying requires clients to run proxy-aware applications. Although most Web browsers and FTP clients can be configured to use a nontransparent proxy, transparent proxies are easier for end users to live with than are nontransparent proxies. Modern application-layer proxies, such as Zorp, are transparent.

Transparent or not, proxying has several important ramifications. First, low-level anomalies, such as strange flags in the IP header, generally are not propagated by the firewall. The firewall initiates the secondary connection in the way that it, not the client system, considers an acceptable manner. Second, because the firewall is re-creating the client connection in its entirety and not merely propagating or trivially rewriting individual packets, the firewall is well positioned to examine the connection at the application layer. This is not a given, however; if the firewall is, say, a SOCKS firewall and not a true application-layer proxy, it simply could copy the data payloads of the client connection packets into those of the new, proxied packets. But if the firewall is application-aware, like Zorp is, the firewall not only examines but makes decisions about the data payloads of all client packets.

Let's look at an example: suppose your public Web server is vulnerable to a buffer-overflow exploit that involves a malformed HTTP GET command containing, say, an abnormally long URL. Your application-layer proxy firewall initially accepts the connection from the client, but upon examining the long URL, closes the connection with an error message to the client and a reset to the server, without ever forwarding the attack payload, the long URL.

The third ramification isn't a positive one: by definition, proxying is more resource-intensive than is packet filtering, and application-aware proxying is especially so. This strike against application-layer proxies is, however, generally overstated. Zorp, for example, can proxy 88Mbps worth of HTTP traffic, nearly twice the capacity of a T-3 WAN connection, running on only a 700MHz Celeron system with 128MB of RAM. Zorp, on a dual-processor Pentium system with 512MB of RAM and SCSI RAID hard drives, can handle around 480Mbps, according to the Zorp Professional v2 Product Description, available at www.balabit.com.

In summary, application-layer proxies provide superior protection by inserting themselves in the middle of each network transaction they allow by re-creating all packets from scratch and by making intelligent decisions on what application-layer commands and data to propagate. They accomplish this based on their knowledge about how those applications are supposed to work, not merely on how their container packets ought to look. The main strike against application-layer proxies is performance, but thanks primarily to

Moore's Law, this shortcoming is mitigated amply by fast but not necessarily expensive hardware.

In the interest of full disclosure, I should mention one other shortcoming that many people perceive in application-layer proxies, greater complexity. It stands to reason that because application-layer proxies are more sophisticated than packet filters, it should take more sophistication to configure them, in the same way that you need to know more to operate a Mosler safe than to operate your typical bus station locker. It's more work to configure a firewall running Zorp or Secure Computing Sidewinder than it is to configure one running Check Point Firewall-1 or Linux Netfilter/iptables.

But isn't better security worth a little extra work? Like everything else in information security, it's up to you to choose your own trade-off. Maybe the extra work is worth it to you, and maybe it isn't. Either way, I hope this column makes you glad you've got the choice in the first place. The remainder of this article, which continues with at least one more installment, explains precisely what's involved in configuring and using Zorp.

Getting and Installing Zorp

The proxy dæmons that comprise Zorp run on top of the Linux kernel concurrently with the standard Netfilter and Balabit-provided TPROXY kernel modules. In theory, this makes Zorp distribution-agnostic, and it's designed to compile cleanly on any Linux distribution that meets certain requirements (see below). Zorp is developed on Debian Linux, however, and the vast majority of Zorp documentation assumes that you're running Debian too. In fact, Zorp GPL is an official Debian package (as of this writing, in Debian's testing and unstable releases).

Zorp is available in three versions: Zorp GPL, the free GPLed version; Zorp Unofficial, a cutting-edge or beta version of Zorp GPL; and Zorp Professional (or simply Zorp Pro), a commercial product based on but with more features than Zorp GPL. If you purchase Zorp Pro, you get a bootable CD-ROM that installs not only Zorp Pro but ZorpOS, a stripped-down Debian distribution optimized for Zorp. With Zorp Pro, a bare-metal Zorp installation takes less than 15 minutes, excluding subsequent configuration, of course. Anyone who's suffered through lengthy dselect sessions while trying to install just enough Debian for one's needs can appreciate the beauty of this.

Zorp Pro also includes the new Zorp Management Server (ZMS), which allows you to manage multiple Zorp firewalls from a central management host. The host in turn can be operated remotely with ZMC, a GUI client available in both Debian Linux and Windows versions. ZMS is functionally equivalent to Check Point Firewall-1's management module, arguably the biggest reason Check

Point has conquered the enterprise firewall world. ZMS has the potential to make Zorp very attractive indeed to sites with a lot of firewalls to manage.

ZMS/ZMC is still a little rough around the edges—Balabit isn't expecting to release a consumer-installable version of that part of Zorp Pro in March 2004 (though at the time of this writing it is being used, successfully, by paying customers). Even if you don't use ZMS/ZMC, Zorp Pro's smooth installation and wide range of features, including several application proxies not supported in Zorp GPL, make Zorp Professional worthwhile.

Unlike Zorp Pro, Zorp GPL and Zorp Unofficial require a working Linux installation that includes the following: glib 2.0, Python 2.1, libcap 1.10 and openssl 0.9.6g. It also requires either a Linux 2.2 kernel compiled with IP, firewalling and transparent proxy support or a Linux 2.4 kernel compiled with iptables, iptables connection tracking, iptables NAT and, using Balabit's TPROXY kernel patch (www.balabit.com/products/oss/tproxy), iptables transparent proxying. All of these features should be compiled as modules.

Once your OS is ready, you either can install Zorp GPL from binary deb packages or compile Zorp GPL from source code (available at www.balabit.com/downloads). Compiling Zorp GPL is a little more involved than your typical ./configure make make install routine; see the Zorp GPL Tutorial at www.balabit.com/products/zorp_gpl/tutorial for detailed instructions.

Next time, I'll describe how to set up Zorp GPL to protect a typical Internet—DMZ—Trusted Network topology.

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant in Minneapolis, Minnesota. He's the author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux for Suits

The Fracturing Desktop

Doc Searls

Issue #119, March 2004

A business work system is not a family entertainment center. Doc points out where the “desktop” market is heading and where Linux fits.

Back in 1997, not long after Microsoft famously bought WebTV for $425 million US, I was approached by Dave Feinleib, then a Microsoft executive, to write a chapter for a book he was compiling on the coming convergence of—naturally—the Web and TV. At first the book wasn't going to be published by Microsoft, but that's what ended up happening. In typical Microsoft fashion (and over Dave's objections), it was given a terrible title: *The Inside Story of Interactive TV and Microsoft WebTV for Windows*. This title was too bad, because it otherwise was a good book, containing little promotional poop about Microsoft products.

My chapter was the biggest one in the book. I'm glad I wrote it, because it forced a lot of thinking that ended up in *The Cluetrain Manifesto* (www.cluetrain.com/book). The concluding paragraphs of my chapter, written almost six years ago, have relevance to the desktop focus of this month's issue:

> This might look like a long shot, but I'm going to bet that the first 50 years of TV will be the only 50 years. We'll look back on it the way we now look back on radio's golden age. It was something communal and friendly that brought the family together. It was a way we could be silent together. Something of complete unimportance we could all talk about.
>
> And, to be fair, TV has always had a high quantity of good stuff. But it also had a much higher quantity of drugs. Fred Allen was being kind when he called it “chewing gum for the eyes”. It was much worse. It made us stupid. It started us on real drugs, such as cannabis and cocaine. It taught us that guns are the best way to solve problems and that violence is ordinary. It disconnected us from our families and

> communities and plugged us into a system that treated us as a product to be fattened and led around blind, like cattle.
>
> Convergence of the Web and TV is inevitable. But it will happen on the terms of the metaphors that make sense of it, such as publishing and retailing. There is plenty of room in those metaphors—especially marketing—for ordering and shipping entertainment freight. The Web is a perfect way to enable the direct-demand market for video goods that the television industry was never equipped to provide, because it could never embrace the concept. They were in the eyeballs-for-advertisers business. Their job was to give away entertainment, not to charge for it. So what will we get? Gum on the computer screen or choice on the tube?
>
> It'll be no contest, especially when the form starts funding itself.
>
> Bet on Web/TV, not TV/Web.

Microsoft bet the other way and lost. What they offered was Web-on-TV, not TV-on-Web. The WebTV product was a device, plus a service, for the tube. It failed in the marketplace. Today the brand is dead, and the webtv.com URL redirects browsers to MSN TV, a service the site calls “a combination of an easy-to-use set-top box and an affordable monthly subscription”. The site has all the energy of a headstone.

Meanwhile, look at what's happening to TV. For starters, TVs aren't tubes anymore. All the best new TVs (soon, *all* TVs) are flat-panel plasma and LCD screens. They're essentially computer monitors, only with bigger pixels and blurrier images, unless they're displaying high definition broadcasts (which are still rare) or DVDs (which are still merely enhanced 640×480 images).

More importantly, TV programs are simply another form of content. That's the lesson taught by new flat-panel monitors that include a TV tuner at little or no extra cost. Right now CompUSA sells a tuner-equipped KOGi 17" 1280×1024 monitor for $399 US (after a $100 rebate). And prices are sure to drop further by the time you read this.

If you want to be more than a desk potato and actually *do* something with your TV input, you can put the tuner where it belongs—inside your computer. CompUSA has TV tuner cards for less than $40 US, and there are plenty of hacks to make them work with Linux. There also are plenty of hacks that let your TV do TiVo-like PVR duty. See Video for Linux (www.exploits.org/v4l) for piles of information on these and other projects.

Although most home desktop machines still are all-purpose devices, the TV convergence that's beginning with monitors is a harbinger of two huge forks that are coming in the desktop computing category. The first fork is one between home and office environments, the other is between fixed and mobile (desktop and laptop) in both environments. The two forks are orthogonal, and the 2 × 2 result is a marketplace that's fracturing into four different categories (Table 1).

Table 1. The New Fractured Desktop Market

	Home	Office
Fixed	Entertainment center	Generic desktop
Mobile	Home desktop replacement laptop	Road warrior notebook or laptop

In this new market the home PC turns into a family entertainment center—or what Steve Jobs calls a digital lifestyle hub—and the office PC turns into a generic desktop. The needs in each environment are so different that there's little reason for the two even to maintain the same form factor. Your home PC needs the backplane and connections to handle USB and FireWire devices, game controllers, coaxial and composite video input from your cable TV or satellite box and so forth. Your office PC needs to be cubicleware: a monitor with a CPU that's connected to the Net and not much more. At home your application suite includes a potentially infinite variety of stuff. At work you need e-mail, a browser, minimal office productivity applications and your company's arcane internal applications. More important, for your company's sake, your cubicle's box needs to be managed, secured and supported easily. All of which are reasons why Linux is getting big, fast, in the generic desktop quadrant. (See this month's *LJ* Index for some of the huge numbers flying around.)

Laptops also are getting far more capable. Not long ago, if you wanted big storage, a large flat-screen or DVD burner, your only choice was a desktop. Now all those features (including screens up to 1600×1200) are available in desktop replacement laptops. Add Wi-Fi, standard now on most new laptops, and your home office can be anywhere. The choice for the road warrior professional is mostly a matter of size. And regardless of size, the trend is clear: the most personal computers today are laptops, not desktops.

In the January 2004 issue, I outlined the challenges that still face Linux on laptops. Since then a number of new Linux laptops have appeared, notably from Lindows and Element. At this moment, Lindows (whose LindowsOS is based on Debian) has a Laptop Edition that addresses familiar power-management and wireless card device driver issues, and it is sold bundled with

other brands, starting at well under $1,000 US. Element is selling a Linux Tablet for $999 US and a notebook for $799 US. More significantly, Mike Hjorleifsson, Element's CEO, tells me his company is working closely with VIA, NVIDIA and ATI to give Linux at least parity performance alongside Windows and Mac laptops that use the same kinds of devices, such as DVD and CD-RW drives. This is a significant development. When I asked Hjorleifsson, “Are you driving a truck through the hole Intel left open when it neglected to release Centrino device drivers for Linux?”, he said, “You've got it.”

It's reasonable to expect the big OEMs—Dell, HP and IBM—to continue protecting the high margins on their Windows laptops for as long as possible by *not* marketing equally functional Linux alternatives. And that's fine, because it leaves the market open for more ambitious players who aren't afraid to discover the market demand we all know is there—and not only in this one category we used to call desktop.

Doc Searls is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

EOF

Lest We Forget, Why Open Source Wins

Chris DiBona

Issue #119, March 2004

Customer experience shows that open source is the greatest insurance policy an IT department can have.

VA Research was, for a time, the best place I ever worked. Shortly after I started there, then-CEO Larry Augustin made it clear that one of the things he wanted me to do was manage the company's relationship with the Linux community. We started by collecting old machines and handing out accounts to developer groups.

Whether it was the Free Software Foundation, Debian or Stampede, we simply wanted people to know they had a place to go when their bandwidth demands exceeded the limits set by their ISPs. This worked pretty well for a while, until the number of machines on which we were hosting projects started numbering around 40 and the extra load started taking its toll on our system administrators. Around this time, Tony Guntharp, Tim Perdue, Uriah Welcome and others locked themselves in a room with a few pallets of SCSI drives and 68 days later, introduced the world to SourceForge.net.

Tony and his colleagues expected about 1,000 people on the site by the end of the year, but they had over 5,000 by the end of that first month. The growth wasn't showing any sign of slowing down. There was pressure from some of the executives and board members to shut down the site due to cost concerns. Luckily for the site and for the future of VA, those concerns were held back by the work of people like Larry Augustin and Steve Westmoreland. The code of the SourceForge site was developed much like many of the projects on the site under the GPL.

Then, VA got a new CEO and went out of the hardware business. GPL releases of the code stopped dead, and installing SourceForge on-site became the

business of the company. Along with this change, the company decided to go proprietary.

The nature of the GPL meant the code base still was out there. After about three months, Tim Perdue, who was part of the original team, went to work and released GForge, a new version for those who were looking to maintain their own or their customers' SourceForge-like sites. At this point, in the interest of full disclosure, I should note that the company Tony Guntharp, Steve Westmoreland and I started, Konstrux Technologies, installs and maintains these kinds of sites for our customers. We base this business on GForge; Tim Perdue and others have similar businesses.

SourceForge.net is a fantastic resource and continues to be; OSDN and VA should be remembered and thanked for that. I don't mean to rail against their decision to go proprietary, as I'm sure they felt they had good reasons to do so.

But they were wrong. From a customer and vendor perspective, sticking with a completely open-source solution is wildly advantageous. I do understand the profits to be gained from proprietary software, and I even believe there are places where proprietary software is likely to remain ahead of open-source software. If you measure the success of a code base by features, however, open-source software is the winner. Since forking off from SourceForge, GForge has added full administrative and installation documentation, significant code and UI clean up, XML interfacing and an installer.

But from the customer or vendor perspective, the advantages of open-source software too often are forgotten. Our company had a customer who needed a role-based authentication system added onto GForge. While we were writing a statement of work for a contract to add this piece, a programmer who had implemented a similar system at Mitre submitted a patch to do 90% of what the customer wanted. We were able to meet the needs of a customer without bringing on more employees, spreading ourselves too thin or charging too much. We thus were able to compete against proprietary vendors who were much larger than us.

From a customer perspective, open source is the greatest insurance policy an IT department can have. In the unlikely event Konstrux should go out of business, our customers wouldn't find themselves with a collaborative development site they can't maintain. This is not something that can be said about our competitors. If they go out of business, that code becomes part of the bankruptcy auction and their customers may be left high and dry. Through open source, customers can protect their investments in information technology. This fact is obvious to open-source software veterans, but I think

that people need to remember this is one of the freedoms with which we're concerned.

The Free Software Foundation tends to speak in terms of individual liberties, but their ideals extend nicely to business, giving companies the freedom to run their own IT departments and to preserve their business systems without being held hostage by hostile vendors or the courts.

Chris DiBona is a cofounder of Konstrux Technologies and was the co-editor of the *Linux Journal* 2000 Book of the Year, *Open Sources*, and an editor for the Web site Slashdot.org. His Web site can be found at dibona.com.

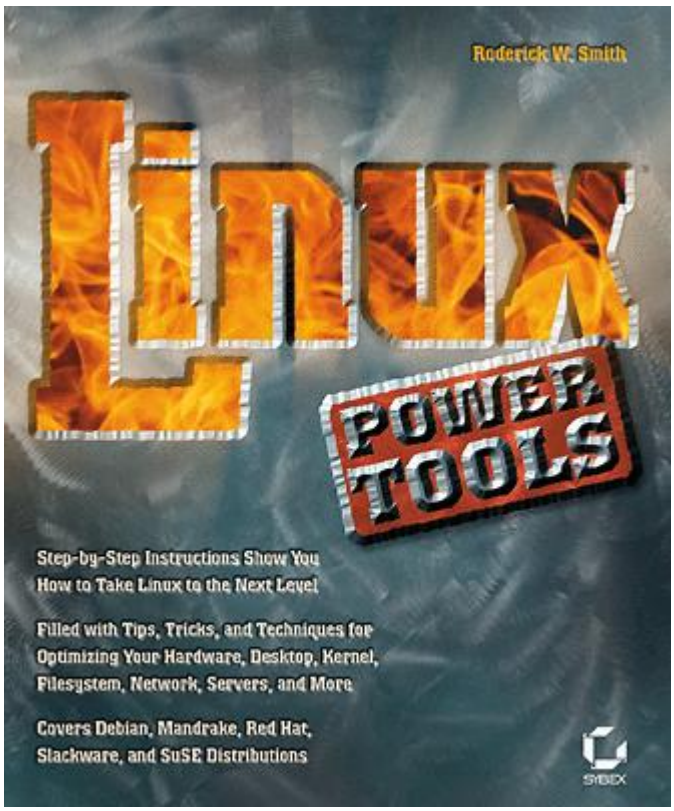
Archive Index Issue Table of Contents

Advanced search

Linux Power Tools by Roderick W. Smith

Suresh Krishnan

Issue #119, March 2004



Sybex, 2003

ISBN: 0782142265

$49.99 US

I have been a UNIX user for more than ten years, and I recently made the move to Linux. When I picked up *Linux Power Tools*, I thought I had nothing new to learn about Linux, but boy was I wrong. After reading this book, I realized how modern Linux is compared to the Unices I have used. *Linux Power Tools* is a

neat how-to book that explains many commonly performed tasks. It is great to read either from cover to cover or as a quick reference. The distribution-agnostic tone is one of the nicest things about this book and should help it find a wide audience.

The order in which topics appear in the book is an accurate portrayal of how a user would build a system in real life. No information is provided about how to install Linux on the system, however, because it's covered in your distribution's installation guide.

This book starts off by offering tips on how to locate drivers for your devices in case they were not detected automatically or were detected improperly. It then moves on to tweaking disk performance parameters and optimally laying out partitions on your hard disk. Shell programming is covered only briefly.

System administrators should love the coverage of both text and GUI system configuration mechanisms for all the major distributions. This book contains one of the best introductions to SystemV startup scripts I have ever read; it is an absolute must-read. If you have wondered why fonts look so ugly in your Web browser, you will learn the causes and solutions in the excellent coverage of X configuration and X font configuration. Recompiling the kernel is a tricky issue for a novice user, but this book effortlessly walks you through the process. Any machine connected to the Internet always is under attack, and the section on building firewalls using iptables is a life-saver. And, when you feel ready to run your own services, *Linux Power Tools* provides you with information to get the most common Internet services, such as Web and mail, up and running in no time.

In addition, *Linux Power Tools* has a good index, making it easy to find specific topics. I found a lot of tips sprinkled throughout the book, such as the -j option for tar to pass the file through bzip2. I would recommend this book if you are a new Linux user looking to make better use of your system or a power user looking for a quick reference book. If you want to excel in shell programming or become a crack sysadmin, though, you should look elsewhere.

Archive Index Issue Table of Contents

Advanced search

EmperorLinux Meteor Notebook

Tony Steidler-Dennison

Issue #119, March 2004

Built around the Sharp Actius MM10, the Meteor is the smallest fully functional notebook I've yet seen.

Product Information.

- Manufacturer: EmperorLinux
- URL: www.emperorlinux.com
- Price: $1,700 US

The Good.

- 2.1lbs, .52" depth.
- Built-in wireless and full connectivity options.
- Easy configuration with installed Red Hat 9.
- Custom kernel featuring software hibernation.
- Focused documentation.

The Bad.

- External CD-ROM only, not included.
- Touchpad placement.
- 1GHz processor.
- Short battery life.

We live in the midst of an industry rife with buzzwords. In many ways, these quick phrases are the coin of the marketing realm; they are words that press some personal hot button, driving us inexorably toward the purchase of the

latest and greatest technological device. They're powerful tools, but the use of these buzzwords often serves only to blur the line between new device types.

Take, for example, the current hot word used to describe mobile computers, notebooks. Every major manufacturer has a notebook in its product line. The word itself, it seems, is evolutionary, derived from the concept of a laptop computer. Notebook says to the consumer, “I'm smaller than an unwieldy old laptop, small enough to take you back to your college days and those indispensable collections of paper bound together with thin spiral wire.”

As is so often the case, the success of the word notebook has led to its widespread misuse. You'd be hard pressed to find a major manufacturer carrying a line of laptops these days. The notebook imagery is too powerful, leaving manufacturers with little choice but to throw the old laptop description in the dustbin. Even if the device weighs in at better than five pounds, it sells better as a notebook than it ever will again as a laptop.

Fortunately, some real notebooks are on the market that serve to clear the confusion. These are devices that provide congruence between the notebook imagery and its reality. Lying somewhere between a PDA and a laptop, notebooks fulfill a critical niche for users weary of lugging the old laptop through airports and hotel lobbies. One device in particular, the EmperorLinux Meteor Notebook, has re-established the descriptive value of the notebook buzzword. With dimensions and weight that rival some of my own college notebooks, the Meteor is Linux-ready and built to travel for any savvy computer user.

Built around the Sharp Actius MM10, the Meteor is the smallest fully functional notebook I've yet seen. Weighing in at a mere 2.1lbs (with battery), it's light enough for even the most wrist-weary mobile worker. With a thickness of .52", the real danger is it may become lost in the soft-sided leather briefcase I've used for years to carry my laptops. The moment I pulled the Meteor from its box, I knew it held real promise to reclaim the notebook buzzword for what it really should be.

In its factory configuration, the Meteor is marketed and installed by Sharp as a Microsoft Windows machine. Filling a valuable niche in the mobile market, EmperorLinux converts these notebooks and replaces the original OS with Red Hat 9. The match of the two components is nearly perfect, providing an extremely usable Linux desktop and application set. The 2.4.2x Linux kernel is custom configured in the EmperorLinux shop to provide such mobile-valuable features as software hibernation. EmperorLinux does provide the Meteor with a minimal DOS installation for legacy users, allowing GRUB to handle the bootloading tasks.

Clearly, some trade-offs are made in the Meteor for the sake of size. I anticipated that its screen and keyboard size might make it difficult to use. Even my current laptop, a Dell Inspiron 1100, has a 14" screen and a keyboard that approximates the size and feel of a desktop computer. Surely, I thought, that look and feel couldn't be replicated on a device so small. But upon investigation, the 10.1" XGA LCD screen is bright and sharp, offering far better contrast than my Dell or many of the other laptops I've seen and used. Running at a resolution of 1024×768, the display is surprisingly easy on the eyes, suitable even for graphics manipulation in The GIMP. With the anti-aliasing support in Red Hat 9, I quickly came to prefer the Meteor over the Dell. The display and feel vs. size compromise turns out to be hardly a compromise at all.

The keyboard, although undeniably tight, retains much of the feel of a laptop. In other words, with regular use, it's quite easy to make the adjustment from desktop to notebook. I made it without a hitch, even with the dexterity of a corn-fed Iowan, manual dexterity that's often compared to that of our primary export—hogs. The lone exception was the location of the touchpad. It took some mental training to avoid tapping the pad with my thumb and unexpectedly launching an application.

On the hardware side, a little more ground is given for the sake of the Meteor's compact size. Although these may present some small aggravations to hard-core coders, my sense is they are not the target market for the Meteor. With a 1GHz Transmeta Crusoe processor, the Meteor does feel perceptibly slower than my regular laptop when compiling and installing applications. The latest release of OpenOffice.org took nearly twice as long to install on the Meteor as it did on the 2GHz Celeron-equipped Dell laptop. However, the Crusoe architecture left little discernible difference in execution speed for most applications. Once compiled and installed, OpenOffice.org seemed to run as easily and as quickly on the Meteor as it did on any other platform in my home. My other killer mobile application, Mozilla, opened and churned through pages and images with the ease of a much more powerful desktop.

With 256MB of fixed DDR RAM and a 15GB hard drive, the Meteor once again has hit the sweet spot for most users. That there's not more RAM or storage space is, ultimately, a fair trade for making this device as small and mobile as it is. I missed neither when trading my daily use of the Dell laptop for the Meteor.

Figure 1. Meteor ports include USB and FireWire.

If you're of the personality type that is inextricably trapped in those marketing buzzwords I mentioned, let me give you a new one to distinguish the Meteor from the current crop of notebooks, ultra-mobile. For starters, the Meteor comes equipped with built-in Wi-Fi. With the installed Red Hat networking tools, it's a simple task to set up a Wi-Fi DHCP connection and start surfing or checking e-mail within a matter of minutes. No hot spot close by? No problem. The Meteor also features a built-in 10/100 Ethernet port or, at worst, a PCMCIA slot into which you can slide a modem card. When you put that diversity of connections in its proper perspective—that is, a device a half-inch thick—the Meteor easily deserves the ultra-mobile label.

The custom kernel configuration EmperorLinux provides unlocks some other great hardware features in the Meteor. The notebook provides FireWire and USB capabilities, with one and two ports respectively. The Meteor also features a unique USB-connected vertical docking cradle. This feature allows a user to share the notebook's hard drive with a desktop system or to sync data between the desktop and notebook with ease, even when the notebook is powered down. With the Meteor off, I placed it in the cradle. This assigned the drive to /dev/sdb. I then was able to mount the drive at /mnt/meteor and the /home partition at /mnt/meteor/home. With this completed, I moved data between the machines effortlessly from the command line. I also completed these tasks by opening multiple instances of Nautilus, in effect dragging and dropping data from one machine to the other.

Figure 2. The Meteor Syncing from Its Cradle to a Desktop

In short, the real strengths of the EmperorLinux Meteor are many. It's highly mobile, with the capability to connect to the network across the full range of options. With Wi-Fi becoming increasingly pervasive, the Meteor/Red Hat combination provides both built-in hardware and easy configuration for connecting to the nearest hot spot. The custom kernel relieves even a newbie user from the pain of unlocking all the built-in hardware features. And the sync/storage capabilities provided by the USB docking station are the quickest path to taking your data on the road.

If those features don't fill your bill, consider the documentation provided by EmperorLinux. Though a thin book, it provides step-by-step guidance for setting up and using the most critical features of the Meteor. Unlike some manufacturer-provided documents, the Meteor documentation is kept current with the version of Linux in use in the Meteor. It also features custom kernel-specific information for those who are technically inclined. The documentation is exactly enough, without being too much.

You might think I found the Meteor to be without flaws, but that's not quite true. The flaws, however, are not show-stoppers. As expected, they're related in large part to the size of the Meteor. There is no built-in CD-ROM, although you can connect one to a USB port. As I've already noted, the touchpad placement is a bit awkward. I've never been a big believer in the mouse nubbin found on

some laptops, but the size of the Meteor would make it a good candidate for such a pointing device. The processor is a bit too slow to push the Meteor into the class of machine necessary for developers and coders. Despite its power-miser Transmeta Crusoe processor, the Meteor sucks down a battery like a script kiddie sucks down a Big Gulp. On average, I could expect less than two hours of battery life before breaking out the power cord. The built-in Wi-Fi always is on, regardless of whether it has a connection, which adds to the power requirements and diminishes battery life. Finally, at $1,700 US, you actually might find the small Meteor a bit bigger than your wallet. Even at that price, it's a good investment of time and money for the enterprise.

Figure 3. Its small size makes the Meteor a true notebook.

So, let's refine our marketing-speak a bit. Laptop does not equal notebook. The Meteor Notebook is proof of that, it being the only Linux-equipped device truly to fill the notebook bill. Call it an ultra-mobile if you must, but the Meteor surely will redefine how you hear the notebook marketing message from now on.

Tony Steidler-Dennison is a freelance PHP programmer and technology consultant who frequently writes about mobile Linux technologies. His Weblog, “Frankly, I'd Rather Not” (steidler.net), covers topics from technology to politics. Tony's other on-line presence, uptime (uptime.steidler.net), focuses on Linux for new users. He's currently writing his first book, *Practical Linux Administration*, and gladly discusses Linux-related topics at tony@steidler.net.

Letters

Readers sound off.

Will DMCA Lawyers Burn Books?

In the From the Editor in the December 2003 issue, Mr Marti mentions the courts decided in favor of “banning our technology journalism colleagues at 2600 from even linking to one DVD-descrambling program, DeCSS.” I cannot help but wonder if this censorship would go so far as to be enforced against paper publications. In other words, would they press this law against a newspaper or magazine (like yourselves) who might decide to print the URL of such a link? It would be interesting if they would push the DMCA to such an extreme as to be clearly contrary to the freedom of the literal press, as opposed to the Web-based “virtual” press.

—
Lisa Corsetti

Maybe if some magazine published the URL www-2.cs.cmu.edu/~dst/DeCSS/Gallery/dvd-hoy-reply.htm, you would find out. —Ed.

ATI Open Source Friendly?

In the December 2003 issue of *LJ*, page 40, Glenn Stone raises the point that the “Big 3” graphics core logic vendors are standing firmly on the side of releasing binary-only drivers for their products. I wanted to mention that ATI provides programming information for most of their RADEON line under NDA to interested developers. Some XFree86 and DRI developers already have access to this documentation, and an open-source RADEON driver is underway.

ATI also provides code samples and support for their products; a Mach64 DRI driver recently was developed from scratch due to their involvement. Neither NVIDIA nor Matrox will provide any programming information, even under NDA, to developers wishing to write open-source drivers for their recent products. ATI, therefore, can be regarded as the one mainstream company that, while understandably not running to open source the drivers it has

developed in-house, is nevertheless a friendly force to open-source developers as well as all users that prefer open source.

I also have another approach to consider. Given that 3-D graphics is still a cutting-edge technology upon which new ground is broken on a daily basis, the OEMs consider it crucial to their business model that their proprietary secrets remain guarded. An all-or-nothing approach is counterproductive. I say that we firmly and reasonably draw a line beyond which further source code or documentation revelation would be of diminishing return, and ask *that* of the companies involved, instead of asking for everything including their crown jewels.

Stone also calls for debate on this topic. I see very little to debate, besides to buy products only from the company that gives you the level of support you desire, whether it be binary-only drivers that work “most of the time”, open-source drivers developed and supported in-house or simply the availability of programming documentation for their hardware. Also, consider the value of whether you wish to participate in a self-supporting user community, or whether you really don't mind being dependent on a single source of support for the product and its associated software.

Furthermore, it is imperative that when you make a buying decision that you contact the sales departments of both the company from which you bought the product, to tell it how its enthusiasm for open source caused you to buy its product, and the company's main competitors, to let them know what your criteria were and why you did not choose their product. Only by making informed buying decisions and providing feedback to the companies involved can we ever expect our preferences to have an impact on the market as a whole. Be active if you want to see change.

—
Ryan Underwood

Linux Vendor Helps Low-Budget Customer

Because Monarch Computer Systems supported the Ultimate Linux Box Project [*LJ*, December 2003], I have been considering them when purchasing a new system. Recently, I had a problem with a system that required me to make a contingency plan in case I could not get it fixed. It turned out that I did fix the system, which had a strange power supply problem. However, Monarch was so helpful to me that I thought the least I could do was commend them on their service. I wrote to Monarch:

> Thank you very much for your help. I borrowed a PS from a friend yesterday and that resurrected my A7V system. So I will not be needing a system right now. This is good for me because our little company isn't doing so well and my salary has been cut by one-third. However, when I get more work or get my salary back, I want to get another system as an upgrade and to make this A7V system a backup. When that time comes, I will be coming to Monarch first! Chris has been super-helpful for me in preparing a low-budget new system. This is even more impressive because he knew from the start that this was not going to be a high-margin sale!

I give Monarch my highest recommendation.

—
Michael George

Linux-Powered Organ

Those of you who know me well also know that I collect automated musical instruments and have a fondness for organ music. Nick Walker found this link (www.eetimes.com/sys/news/OEG20031203S0032) and sent it to me about an Aeolian-Skinner organ at Trinity Church in NYC that was destroyed by the dust and dirt caused by the World Trade Center collapsing on 9/11. From there I read several articles, including the one at this address: www.organpower.com/DoubleOpen5401.pdf. On page 6 of the PDF it has the words:

> The decision was made to develop the organ's control and tone generation systems in a Linux environment. Linux, a highly stable operating system, is considered by many to be the finest environment in which to run this type of standalone application on either standard or embedded controller PCs.

The passage is from an article entitled “Epiphany on Wall Street”, *Open: A Publication of the New York City Chapter of the American Guild of Organists*.

—
Jon “maddog” Hall

Why USB Devices Mount Read-Only

In the December 2003 issue, an article by Rick Moen on the use of USB Flash storage devices (“Floppies for the New Millennium”) includes the following: “...no matter what you do, the Flash disk always mounts read-only....Exactly why /bin/mount insists that the Flash disk is write-protected and must be mounted read-only is a genuine mystery.”

I am the author and maintainer of the driver that talks to this entire class of devices. To me, it's not a mystery. USB mass storage devices are handled by a virtual HBA in the SCSI subsystem. For direct-access type devices (disks, Flash, etc.), sd.c checks the write-protect status with a MODE_SENSE command. The problem is, MODE_SENSE isn't used by Microsoft Windows. Thus, the firmware to recognize, process and respond to that command properly often is lacking from USB devices, especially those that are under extreme price pressure, such as the keychain-type Flash devices discussed in the article. Sometimes, they simply respond with incorrect data.

This problem is widespread. I've even seen quite a few devices that crash their internal firmware when they see certain MODE_SENSE or MODE_SENSE_10 commands. This is basically a matter of poorly implemented devices—the device is tested with a popular OS and not fully coverage-tested for compliance to the open and published specifications.

Up to a certain 2.4.x kernel version, a failure of the check for write-protect would default to write-protected status. Often this is accompanied by a message from the kernel (`sd.c: test WP failed, assuming write protected`). Later, a patch was merged that made the assumption write-enabled. The 2.5/6 kernels introduced a new problem—sd.c wants to check for mode page 8, which causes more devices to die. Currently, those of us involved with USB storage development are trying to identify what MODE_SENSE/ MODE_SENSE_10 commands are used by Microsoft Windows to speak to these devices—the theory is that if we can identify these commands, then they must be safe for Linux to use.

—
Matthew Dharm
Author/Maintainer, Linux USB Mass Storage Driver

Safely Allowing Root Access

As a newbie in Linux development, I was delighted to read the article “Controlling Hardware with ioctls” by Lisa Corsetti [January 2004]. So I downloaded the source code and discovered that the ioctl calls used work only if you are running as a privileged user, which I was totally unaware of. In the context of the article, this worked great, but it would be nice if a similar piece of code had been shown to get the same information for an ordinary user.

—
Jeffrey Goddard

When you need ordinary users to do a task that's reserved for root, use sudo: www.courtesan.com/sudo. —Ed.

Photo of the Month: We're Historic

I recently asked the Computer History Museum (www.computerhistory.org) if they would like my back copies of *Linux Journal*. Museums always are reluctant to take items, as it is hard to raise money to take care of the things. So, it was great to hear they were excited to add *LJ* to their collection. I thought you might like to see *Linux Journal* becoming a part of history; here is museum curator Sharon Brunzel holding *LJ* #1. Note that she is wearing curator's gloves, required when handling all historical objects.

—
Pardo

Classified Ads, Please

I've been subscribing to *Linux Journal* for three years now. Great magazine, but I noticed you don't have a classified section. Many readers can benefit from small, low-cost classifieds. I advertise my group in a local computer paper's classified section and was disappointed to see *LJ* doesn't have any classifieds. A section for user groups might be a good addition or simply “help wanted” and “education and training”. Linux thrives on user group participation.

—
Rick Tomaschuk
President, Toronto Area Novell User Group

More on Controlling Devices, Please

Jason Ellison's article “Controlling Devices with Relays” in the December 2003 issue was a refreshing complement to the other articles on kernel scalability. I have been using Linux for almost ten years, but I still consider myself to be both a novice administrator and programmer. Jason's article was perfect for all of us who don't spend our days tweaking operating system internals, but who know just enough about our computers to be dangerous.

In upcoming issues, I would enjoy seeing more articles that take this cookbook approach to describing the diversity of Linux solutions—defining a short problem, reviewing the options available and discussing the solution with enough detail that a sufficiently motivated novice can ring a bell or two at home if they desire.

I work as a mechanical engineer. As part of my job, there are countless opportunities to implement solutions similar to the warning bell system that Jason created. Combining computer control with mechanical hardware can save tremendous amounts of money. The ability to make use of inexpensive components, existing computer equipment and validated Linux software, as opposed to purchasing an overpriced supervisory control and data acquisition (SCADA) package should be applauded.

—
Joe Stevenson

Radio Observatory Uses Linux

The Jodrell Bank Observatory appears to be using Red Hat Linux in connection with looking for a signal from the European Space Agency's Mars Lander, Beagle-2. Look in the lower-right corner for a “waterfall display” (time vs. frequency) plot of the spectrum around Beagle-2's center frequency. The desktop is clearly GNOME on Red Hat. The signal of interest is 5 watts from about 98 million miles away (www.jb.man.ac.uk/public/Thursday3.jpg).

—
Steve Eitelman

New Site for Slashcode Webmasters

Installing slashcode, the content management software that runs the popular Internet site Slashdot (www.slashdot.org), always has been a difficult and arduous task for even the hardened administrator. In response to this, in June of 2002, I created the “Install Slash For Dummies” document, very much like *Linux Journal*'s own slash installation guide (/article/6674). However, the

document did not provide a comment system nor did it provide a community-based forum. In response to this, I have created www.installslash.org. This is a new community site, and its goal is to make it as easy as possible to install slash, and at the same time provide support and valuable information on modifying slash to your liking, no matter at what skill level you reside.

—
Terry Vaughn

Can US Federal Employees Contribute to GPL Projects?

Something that I have yet to see handled in the Open Source community is the difficulty that federal employees have in contributing to open-source/free software. I'd love to contribute more and help out with projects that I use at my (federal) workplace, but I understand that the law prohibits that. Federal employees cannot hold or give copyright in any code created by them, but the code must be public domain. This makes contributing difficult at the very least. I believe that some GPL or other license projects have significant federal-employee written code, but I do not understand how they can place it under the GPL. I'd love to see a treatment/explanation of this. I have not yet seen any. If something does exist, please point me to it.

—
Andrew Gilmore

Any project, under any license, can accept public domain code. The public domain is not a license; see /article/6225. —Ed.

Archive Index Issue Table of Contents

Advanced search

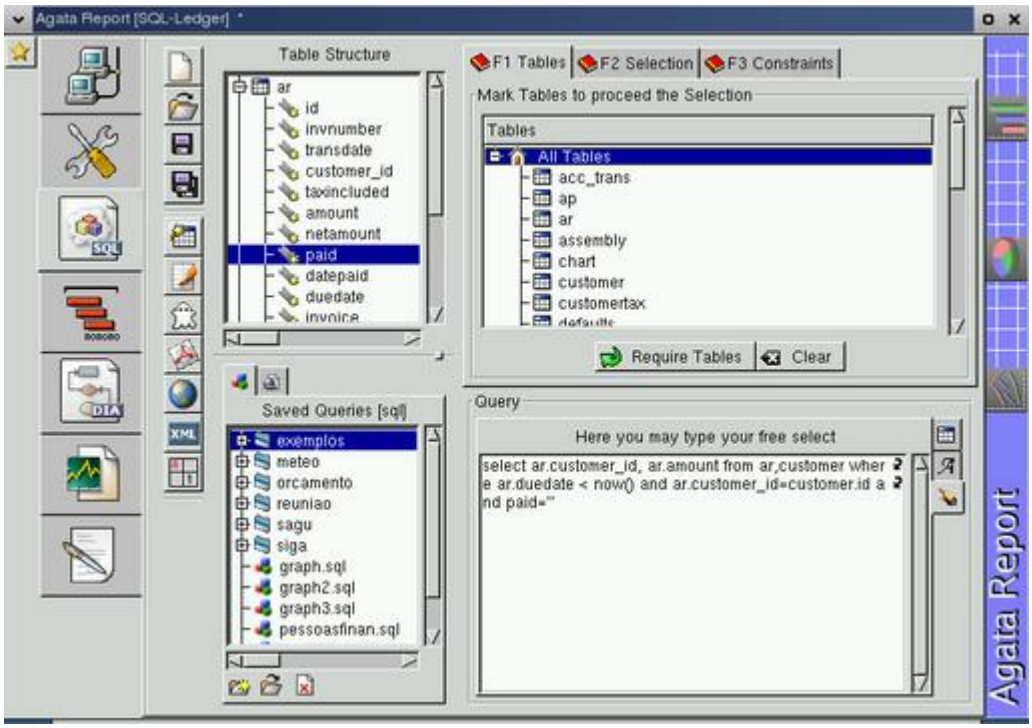
UpFront

Agata: www.agata.org.br

David A. Bandel

Issue #119, March 2004

Agata is a PHP application that runs directly in PHP, not in a Web server. Why? I have no idea, but it works. And, if you need something that resembles (so I'm told) Crystal Reports in Windows, take a look at Agata. It can be used against most SQL databases, although I tested it only against PostgreSQL. Agata is easy to install and run and is extremely flexible. It is also pretty and does graphs. Requires: PHP and php-gtk.

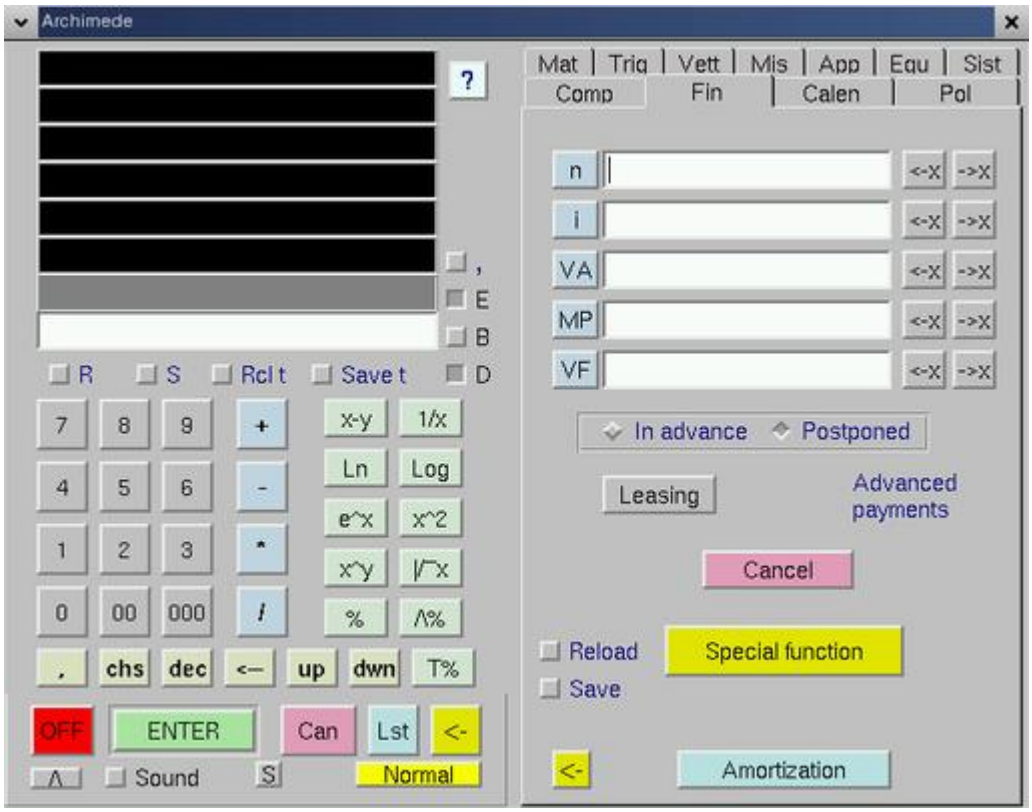


Archimede: **mcz.altervista.org**

David A. Bandel

Issue #119, March 2004

Many years ago in college I had a completely programmable calculator. Last time I saw it, the calculator was covered with dust and I couldn't find a battery for it, so I started using spreadsheets. Now, it looks like I can have a calculator again whenever I want it, at least on the computer screen. Archimede has functions for nearly everything including financial calculations, trigonometric and math functions as well as calendar functions. Requires: libX11, libpthread, libdl and glibc.



diff -u: What's New in Kernel Development

Zack Brown

Issue #119, March 2004

For several 2.6 test releases, **Linus Torvalds** had been hinting that a handoff to **Andrew Morton** was imminent. In fact, as of 2.6.0-test11, it's official. Andrew is now the 2.6 maintainer, even before the official 2.6.0 kernel is released. An interesting facet of this is that the 2.6.0 release is a major event, and Linus is giving Andrew the spotlight. This fits neatly with Linus' response to statements that he is indispensable. Linus always has maintained that he is “just another developer” and not the absolute focal point folks have made him out to be in the past. By stepping aside, Linus may be trying to minimize his public role.

In theory, the purpose of the stable kernel series is to approach true stability gradually. In practice, features from the ongoing development series are often back-ported to the stable series, after some testing. In some cases, the development work is done directly in the stable series, if the changes are isolated clearly and unlikely to cause problems outside of their own small realm. But in general, the ultimate aim of pure stability tends to grow stronger and stronger. And now, with the handoff of the 2.6 tree to Andrew Morton, **Marcelo Tossatti** has decided to clamp down on new features going into the 2.4 tree, and aside from a few possible exceptions like **XFS** support, 2.4 quickly is becoming a “bug-fix and security-fix only” tree.

Pontus Fuchs is working on what to some might seem a radical concept. He is attempting to get certain Microsoft Windows drivers to work with the Linux kernel. Considering that the drivers were written for a completely different operating system, he's actually had a surprising amount of success with this project. Specifically, certain **wireless LAN** cards have no published specifications or Linux drivers, either free software or binary-only. So far, he's been able to use his Broadcom 4301 successfully, and **Pavel Machek** has gotten a Broadcom 94306 working as well.

Carl-Daniel Hailfinger, **Manfred Spraul** and **Andrew de Quincey** have reverse engineered the nvnet driver for the NVIDIA nForce MCP Ethernet adapter and written their own GPLed driver. Actually, to preserve a clean-room development environment that would preclude the possibility that any part of NVIDIA's driver would be copied into their GPLed version, Carl-Daniel and Andrew did the reverse engineering and documented the hardware, while Manfred wrote the new driver from their specifications. In this way, Manfred was guaranteed to have no direct knowledge of the internals of the original driver. Reverse engineering is a time-honored computer-science technique to achieve interoperability, although there is a strong movement to make certain kinds of reverse engineering illegal in many countries. The DMCA (Digital Millennium Copyright Act) is an example of legislation that attempts to control reverse engineering, but what its true impact will be is still being hashed out in the courts.

Some **signal handling** behavior has changed between kernel 2.4 and 2.6. Typically, there are some signals (called thread-synchronous signals) that threads are unable to block. In the 2.4 kernel, a thread attempting to block one of these signals would fail, and the signal would come through anyway. This is not necessarily the most desirable behavior, because there is no legitimate reason for a thread to try to block one of these unblockable signals. In the 2.6 kernel, therefore, when a thread tries to block a thread-synchronous signal, it dies. So the programmer sees right away that something was done that shouldn't have been done. Linus says the new way is the best way, but of course, many past “best ways” have led to even better ways. The same may happen here.

Open eGov

Doc Searls

Issue #119, March 2004

In democracies, the most interesting politics are the kind you can describe with sports metaphors. That's even true when the big story is about technology. This

is what we see in Molly Ivins' line, “the Internet is to politics what television was in the 1960 Kennedy-Nixon race”.

Yet democracy isn't only about elections. It's also about governance. And that's where Linux and open source quietly have been making huge changes over the last year or so. Here's a list of four related initiatives that radically are changing the rules and methods of technology procurement and implementation in government:

- The Open Source Software Institute (OSSI, oss-institute.org) is on a mission “to promote the development and implementation of open-source software solutions within US federal and state government agencies and academic entities”.
- OSSI is working on a US National Institute of Standards and Technology (NIST) validation for OpenSSL, the robust, commercial-grade, full-featured and open-source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols (oss-institute.org/fips-faq.html).
- Project Leopard (leopard.sourceforge.net) is an “eGovernment Web services platform based on LAMP (Linux, Apache, MySQL, PHP/Perl/Python)”. Project founder and *Linux Journal* contributor Tom Adelstein says Leopard is a huge time and hassle saver. “All you have to do is write to this module installer that we have. So, you basically can do the Web pages and database schema and you're done.”
- The Open Government Interoperability Project (ogip.org) produced Project Leopard and is working on the Open Government Interoperability Standard (OGIS), “an initiative to develop an open specification for ensuring that governments and administrative software applications work together more effectively”.

“These efforts are still new”, says Tom, “but already there is a rewarding sense that their implementation is inevitable. The cost is so low, and the rewards so high.”

LJ Index—March 2004

- 1. Estimated thousands of US municipalities that are ripe for open-source applications: 88
- 2. Minimum estimated dollar savings by the US Navy on one open-source interoperability project: 300,000
- 3. Thousands of Linux client installations planned for deployment by Sherwin-Williams Co.: 10

- 4. Thousands of projected Linux-based Sun Java Desktops in a rollout in the UK: 800
- 5. Minimum millions of Linux-based Sun Java Desktops projected for deployment per year in China: 1
- 6. Millions of Linux-based Sun Java Desktops for which Sun is “aiming” in China: 500
- 7. Percentage range of municipal IT budgets “eaten up by productivity suites and tools”: 60–70
- 8. Projected dollar costs to municipal budgets of Linux-based open productivity suites and tools: 0
- 9. Percentage of CIOs and IT managers reporting Linux server use in Australia and New Zealand: 32.4
- 10. Year-over-year Linux server shipment share percentage growth: 51.4
- 11. Income generated in millions of dollars by Linux servers in the last quarter (before November 2003): 743
- 12. Percentage of the server market share for Linux by 2007 at the latest: 45
- 13. Percentage of small companies currently testing Linux: 25
- 14. Percentage of small companies that hope Linux will replace Windows as their core OS: 50
- 15. Percentage of the increase in spending on Linux server shipments between July and September 2003: 16
- 16. Percentage unit increase in Linux server shipments between July and September 2003: 32
- 17. Number of persons supportable by one technical person at Hill House Hammond, before Linux: 50
- 18. Number of persons supportable by one technical person at Hill House Hammond, after Linux: 500
- 19. Minimum percentage of spam relayed off home computers whose owners are unaware: 33

- 1, 2, 7, 8: Open Source Software Institute
- 3: *LinuxInsider*
- 4: *The Register*
- 5: CNET Asia
- 6: vnunet, Sun executives
- 9: CNET
- 10, 11: Ask Web Hosting, IDC
- 12: *The Inquirer*, Meta Group
- 13–18: ZDnet/UK, sourcing IBM and IDC studies and reports

- 19: *New York Times*, sourcing Sophos

They Said It

We're going to immediately roll out the Java Desktop System to between a half million and a million desktops in 2004....It makes us instantaneously the No. 1 Linux desktop player on the planet.

—Scott McNealy, *Business Standard* (www.business-standard.com/ice/story.asp?Menu=119&story=29323)

Linux servers have demonstrated six consecutive quarters of year-on-year revenue growth, proving that they are not a flash-in-the-pan technology and that they are meeting real-world computing requirements in HPC and commercial deployments.

—Jean Bozman, IDC (www.askwebhosting.com/story/25/IDC_Linux_Server_Growth_is_Nearly_50_Percent_Year-Over-Year.html)

Future wide-scale implementations of Linux-based, mission-critical business applications may get less press, but they will continue to happen at all of the largest global financial institutions....Critical to this migration will be the ongoing support of the large hardware vendors, especially their ability to provide support tools and services for porting and new development. We believe that within the next 18–24 months, installing a Linux-based software package will become as normal as installing a Windows or Sun Solaris system.

—Damon Kovelsky, IDC (www.internetnews.com/ent-news/article.php/1559661)

There are no good excuses for binary modules. Some of them may be technically legal (by virtue of not being derived works) and allowed, but even when they are legal they are a major pain in the ass, and always horribly buggy.

I occasionally get a few complaints from vendors over my non-interest in even *trying* to help binary modules. Tough. It's a two-way street: if you don't help me, I don't help you. Binary-only modules do not help Linux, quite the reverse. As such, we should have no incentives to help make them any more common than they already are.

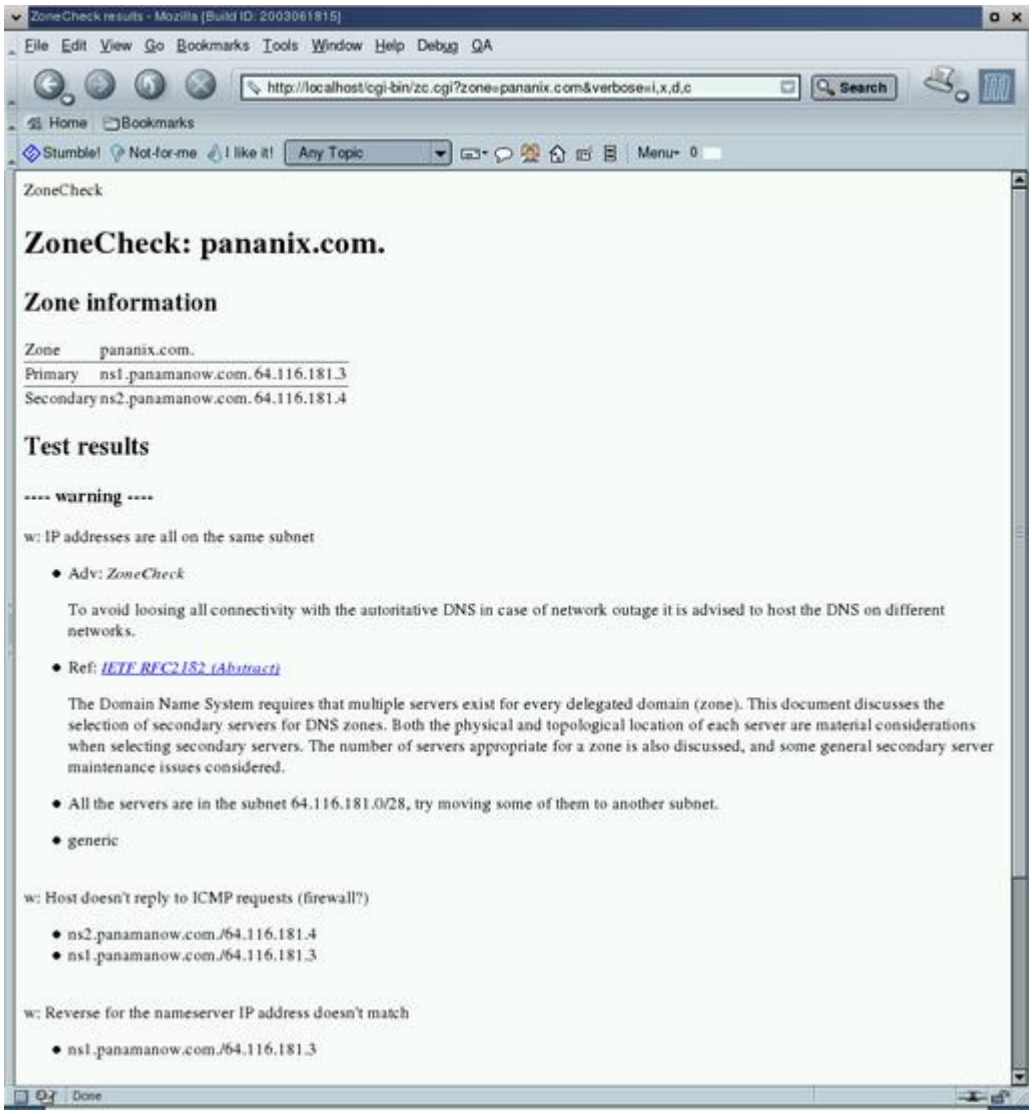
—Linus Torvalds, on the linux-kernel mailing list

ZoneCheck: www.zonecheck.fr

David A. Bandel

Issue #119, March 2004

If you don't remember all the DNS RFCs, or if parts you should know slip your mind, use ZoneCheck to look over your name server zone files to see whether you've maintained your zone as the RFCs suggest. I had forgotten, for example, that the expire time must be seven times greater than the refresh. I had only four times on my zone when I checked. Some 105 different checks are applied. It can be run from the Web or the command line. Requires: Ruby, Ruby extensions yaml, exml and a Web server that supports CGI (optional).



Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Our experts answer your technical questions.

Shutdown Doesn't

I have a dual-OS machine running Microsoft Windows XP and SuSE 8.2. When I do a shutdown in XP my PC powers off. But when I turn off this computer in Linux, my PC does a reboot.

—
Andre Bouve

andre.bouve@pandora.be

This is most likely due to inherent problems between SMP and APM. The two standards are mutually incompatible, apparently resulting from an unavoidable race condition among the multiple processors. In Linux's case, APM is disabled in SMP kernels, even if those happen to be running on single-processor machines. You might try switching to the UP (uni-processor) kernel, or you could compile your kernel with an option that forces the APM power-off feature to work.

—
Jim Dennis

jimd@starshine.org

Try passing `apm=power-off` to the kernel at boot time.

—
Usman S. Ansari

uansari@yahoo.com

Support for ISA Sound Card

I'm having a problem configuring my sound card for my Red Hat 7.2 system. When I run `sndconfig`, it comes back that I have an ISA PNP card, a SoundBlaster 32 Wavetable card to be exact. When it goes to test the sound I get these errors:

```
/sb.o : init_module: no such device
/sb.o : insmod
/sb.o : failed
/sb.o : insmod sound-slot-0 failed
```

When I run `dmesg` I get this:

```
sb: No IsaPnP cards found, trying standard ones...
sb: I/O, IRQ, and DMA are mandatory
No detected device
```

I get that same message when I try any different SoundBlaster card in the list.

—
Joseph Helton

hteam1@mindspring.com

It sounds like your card is configured with I/O base address, IRQ (interrupt request) and DMA (direct memory access) settings that the kernel can't autodetect. You might have to add a line to your /etc/modules.conf that looks something like:

```
option sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
```

where you replace the numbers with those that your card is using.

—
Jim Dennis

jimd@starshine.org

Keeping Bandwidth Bills Down

My broadband ADSL Internet connection has a monthly fixed fee if downloads don't exceed some maximum limit; at the time of this writing, the limit is 3GB. I'm using PPP over Ethernet to connect to the Internet. I want to know if there is some application that can let me have an account of the transferred bytes over my connection.

—

Guillermo Gimenez de Castro

guigue@craam.mackenzie.br

There are several. ipac is the IP accounting package. MRTG is the multirouter traffic grapher, which may be overkill for your needs, as it graphs usage rather than simply totaling it. You also can use the ifconfig command and look at the received (RX) and transmitted (TX) bytes. All in all, ipac is probably the simplest package for you to examine. See www.daneben.de/ipac.html.

—
Jim Dennis

jimd@starshine.org

Also, the sar command can provide you with several statistics about your network interfaces; as root try the command:

```
sar -n FULL
```

which will provide you with transmitted/received packets and bytes among other information in a timed table. Do a `man sar` for further information.

—
Felipe Barousse Boué

fbarousse@piensa.com

Slackware on Serial ATA System?

I am trying to install Slackware 9.1 on a new system that uses a serial ATA drive. The system uses an Intel motherboard with two two-device IDE ports and two serial ATA ports. I can have a total of six devices set up with this rig. The two IDE ports have two CD-ROM drives and a Zip drive attached to them. One SATA port has the only hard drive, the other is unused. I am booting from Slack's installation CD, and the boot proceeds normally until it gets to the point where the drives are discovered. The system sees all of the ports and sees the SATA drive as hde on IDE2. I have partitioned it so that hde4 is the partition that I want Slack to recognize. The system knows the drive is there, but it stalls at a point with this message `hde4: loading IDE drivers`. I can't get any further than this. What should I look for in dealing with SATA drives on Linux?

—

Ren Colantoni

colanton@lacitycollege.edu

You will find yourself doing some tweaks depending on the hardware and distribution you have. A good starting point to find out more about this is deploylinux.typepad.com/main/2003/07/linux_sata_supp.html. There you will find tips specially related to incompatibilities of controllers, drives and Linux.

—
Mario Bittencourt

mneto@argo.com.br

Best Software for Backup?

After I picked up my first copy of *Linux Journal* on the newsstand, I couldn't believe that I'd found in one issue the answers to several of the problems I've been up against. I immediately bought that issue and subscribed on-line as soon as I got home. After digesting the information on Nagios, I'm now looking to replace my company's current DLT backup solution. Our current environment runs a Windows/Veritas Backup Exec 8.6 solution and we're paying a hefty price for these systems. My question is this; is there a viable Linux solution that supports a wide variety of tape backup hardware? Currently I have four single DLT 15/30GB drives at the office, but I also need to support a seven-tape DLT autoloading library on my home LAN. I'd like something that doesn't require a huge investment in time to learn. After all, it only takes the average user about two clicks of the mouse to lose a file, so I'd like to be able to restore it as easily.

—
Eric Patat

epatat@charter.net

BRU (Backup and Recovery Utility) is reasonably well regarded. It's proprietary but not very expensive. More information can be found at www.tolisgroup.com. BURT (BackUp and Recovery Tool) is at the University of Wisconsin, www.cs.wisc.edu/~jmelski/burt, just as AMANDA was created at the University of Maryland.

—

Jim Dennis

jimd@starshine.org

Two other popular backup programs are Amanda, at amanda.org, which is free, and Arkeia, at arkeia.com, which is proprietary.

—
Don Marti

info@linuxjournal.com

Backup software isn't necessarily the best reason to choose Linux, not because it isn't available, but because it's often the same product. Most of the major commercial vendors of backup solutions now support Linux. There are also mid-range solutions that are more cost-effective but still provide graphical wizards and management interfaces. If you would prefer an open-source solution, there is a wide variety of these options available as well, but you also could simply rely on good old tar and gzip or something more robust, such as cpio. You will need the magnetic tape tools package, mt, and the appropriate driver(s) installed in your kernel. If you do go with a tape library, you also may need to search around for a utility that controls the media loader on the device, so you may want to do some research ahead of time before you buy one.

—
Chad Robinson

crobinson@rfgonline.com

We have found that using our own scripts (mostly in Python) for backups (local and distributed), backup verification and validation and restores has been the best alternative so far for the different backup needs we have. We do perform backups into tape devices, CD-based technologies and into other physical hard disks as data and disk backup. A couple of references: www.linux-backup.net has various pieces of information regarding backups in Linux; also look at the book *Unix Backup and Recovery*, which *Linux Journal* reviewed a while ago. Although the book is a bit old, it may still be worth reading. The *LJ* review is at /article/3839. On the hardware side, check the site www.linuxtapecert.org.

—
Felipe Barousse Boué

fbarousse@piensa.com

Best Tool for a WordPerfect Expert?

I've been using UNIX in some form or other for over two decades. Using Red Hat, I put out camera-ready copy for my latest book, *The Economy and Material Culture of Russia*, 1600–1725—668 pages, larger format, with 104 graphs produced by Stata from 108,000 records in filePro16. My exceptionally handy word processor was WordPerfect for the camera-ready copy. Now the university is forcing me to upgrade my computer, which will have Red Hat Linux 9 on it. My understanding is that Corel no longer maintains WordPerfect, which won't run on Red Hat Linux 9. What is the most suitable word processing package for this project? What do you recommend?

—
Richard Hellie

hell@midway.uchicago.edu

There are many word processors available, and your choice of them depends on your publishing needs. You should begin by examining the ever-present Emacs and the LaTeX and SGML document description languages. Most people find that these are too obfuscated to suit their needs, but it's always worth the examination as these are extremely powerful document layout products once you know how to use them. If you prefer a WYSIWYG word processor, you can install OpenOffice.org or KWrite, both of which are open-source products. Or, if you need better compatibility with Microsoft Office users, you can try either Sun's StarOffice product, which is OpenOffice.org with additional fonts and commercial support, among other things, or IBM's Lotus SmartSuite, which is also a commercial product. These are only a few of the options available, and these options do not even include the desktop publishing products. Take a look around—you might be surprised at the variety of options available.

—
Chad Robinson

crobinson@rfgonline.com

Like much unsupported proprietary software, you can keep WordPerfect going by installing old versions of libraries (linuxmafia.com/wpfaq). If you want to keep the ability to import Microsoft Word documents, you need to apply

another fix, too: www.linuxjournal.com/article/5655.

—
Don Marti

info@linuxjournal.com

Hide That Password!

I know I can't be the only one to exhibit this embarrassing behavior on a semiregular basis, so here goes. For whatever reason, there have been times where I inadvertently disclosed a sensitive password on the command line, mistakingly thinking that my input was going to the stdin of a different program such as ssh or smbclient. I use the bash shell, so this means my carelessness gets written to a history file. Normally, this isn't too big of a problem, but sometimes I end up using a shared account on the system. Needless to say, whoever else has access to this account ends up being able to view my password in the history file. Is there an easy way of telling bash to discard entering a prior or specific entry into its history? I'd rather not have to edit the history file manually, which seems to be the only way I know to cover my tracks.

—
Chris DeRose

cderose@deroseandslopey.com

First, if you realize your mistake before you press Enter, simply press Ctrl-U. Doing this erases all of your typing on the current line. This works at the shell prompt (most Bourne-compatible shells), the login prompt and even in vi (while still in insert mode). If you've already pressed Enter, then your fastest, easiest recourse is simply to re-read the history file that's already on the disk. Since the history normally is written only on logout, this will overwrite the in-memory history. Type `history -r ~/.bash_history`. Of course, this also will wipe all of the other entries from the current session, and it will be as if you just logged in (as far as your history goes).

—
Jim Dennis

jimd@starshine.org

Do a `man history` to check on the options of the history command:

```
history [n]
history -c
history -d offset
```

With no options, history displays the command history list with line numbers. A numerical argument of n lists only the last n lines. The -c or -d options, if supplied, have the following meanings: -c, clear the history list by deleting all the entries and -d, offset Delete the history entry at position offset.

—
Felipe Barousse Boué

fbarousse@piensa.com

Debian Install for SATA Drives?

Is there a Debian-based distribution that would allow me to install on an SATA hard drive? The hard drive controller is a Micro-Star International RAID Bus Controller, using ata_via per hardware identification by YaST, and/or a VIA 8237 per the MSI KT6 Delta mainboard manual. I want to switch back to Debian, but the installer does not recognize that my system has a hard drive. The same goes for using Knoppix's knx-hdinstall. I understand that I can install a live system on an IDE hard drive, add the modules necessary to get SATA support working, copy the whole thing over to the SATA drive and then run LILO to get the system working on the SATA drive, but that sounds a bit too complicated for someone as lazy as I am.

—
Nathan Oliphant

nathan@oliphantparts.org

Well, you can install Debian on just about anything if you bypass its normal installer and use the debootstrap package. There are some tricks to using that, however. On my Wiki pages I have described a technique for installing Debian onto a set of disks under LVM (logical volume management), using nothing but an LNX-BBC (www.lnx-bbc.org) and my network connection. So, if you have a rescue disk like the LNX-BBC that can see and access the SATA hard drives, you could follow basically the same procedure that I describe on my pages (www.starshine.org/sysadmoin/DebootstrapInstallation). I will warn that this is not easy. It is somewhat laborious and my step-by-step description doesn't go into much explanation. It assumes expertise in partitioning (using fdisk),

making and mounting filesystems.

—
Jim Dennis

jimd@starshine.org

Xandros Desktop OS (xandros.com), LindowsOS (lindows.com) and Libranet GNU/Linux (libranet.com) are all Debian-based and maintain hardware compatibility lists. You can look up your Serial ATA hardware on their Web sites.

—
Don Marti

info@linuxjournal.com

Archive Index Issue Table of Contents

Advanced search

New Products

JMP 5.1, Centrus Business Geographics Suite, AccuPoll and more.

JMP 5.1

Version 5.1 of JMP contains a Linux port of the desktop statistical analysis tool. JMP 5.1 can be used to link statistical analysis dynamically with graphics for data visualization. Modeling options are included to help users find root causes of problems when there are multiple variables with nonlinear relationships, when no models are identified and when underlying factors are not measured in data. JMP includes Six Sigma, traditional and custom design of experiment (DOE) tools. New statistical platforms are available for version 5.1 that enable analysis of closely related data points as well as large amounts of data. JMP 5.1 supports Red Hat, Red Hat Advanced Server, SuSE, Mandrake and UnitedLinux distributions.

SAS Institute, Inc., JMP Software, SAS Campus Drive, Cary, North Carolina 27513, 877-594-6567, www.jmp.com.

Centrus Business Geographics Suite

Group 1 Software announced upgrades to all of the products in its Centrus business geographics suite. Centrus products incorporate embeddable technology and a wide array of third-party data sources to solve operational problems in which location is critical. The Centrus GeoStan system, which corrects and standardizes address data with spatial information, now outputs vendor segment identifiers linked to data from leading vendors. It also outputs block suffixes using the US Census Bureau's TIGER 2002 data. Centrus AddressBroker has been architecturally improved to handle running on multiprocessor Linux machines. In addition, all Centrus products now run on Red Hat Linux.

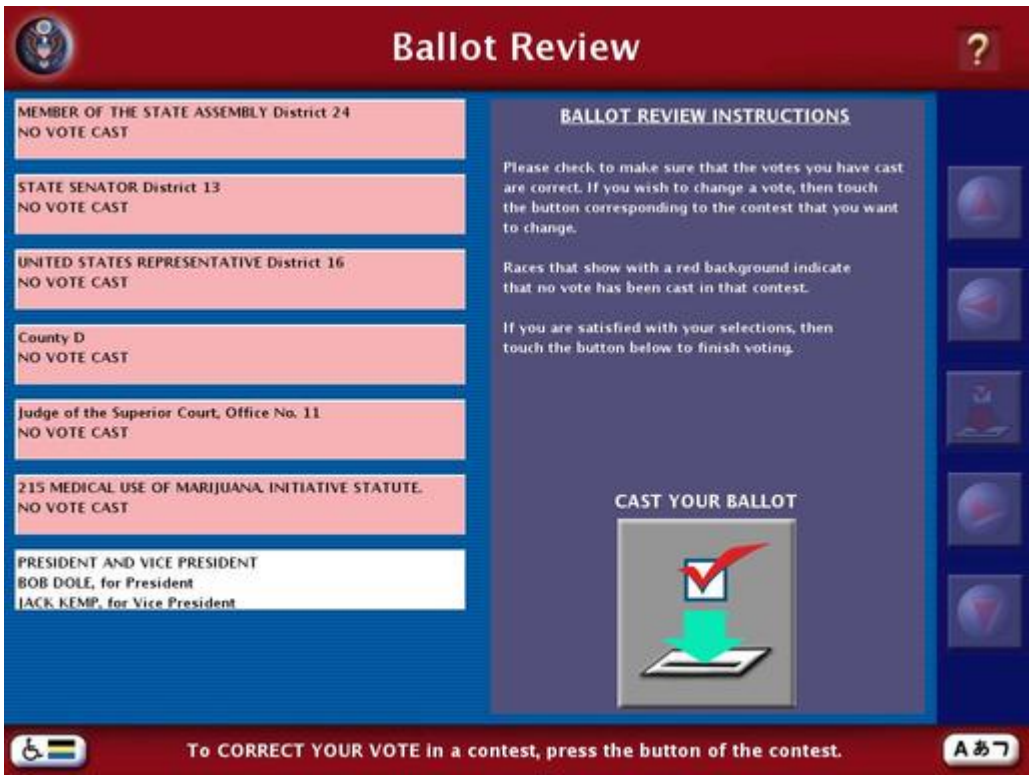
Group 1 Software, 4200 Parliament Place, Suite 600, Lanham, Maryland 20706, 888-413-6763, www.g1.com.

GROUP 1 SOFTWARE

AccuPoll

AccuPoll released its new multilingual electronic voting system, which combines the transparency of touchscreen input with the documentation of a voter-verified, printed paper record. The voting station guides voters through the voting process, and votes are confirmed with an on-screen acknowledgement and a paper Proof of Vote printed by the voting station. Once the vote is cast, the AccuPoll system provides an independent, voter-verified audit trail that is recorded simultaneously in multiple locations in both paper and electronic formats. The AccuPoll system meets the requirements of the Help America Vote Act of 2002, as well as accessibility requirements for disabled voters. AccuPoll runs on nonproprietary hardware and open-source software.

AccuPoll Holdings Incorporated, 15101 Red Hill Avenue, Suite 220, Tustin, California 92780, www.accupoll.com.



VxWorks Compatibility Layer

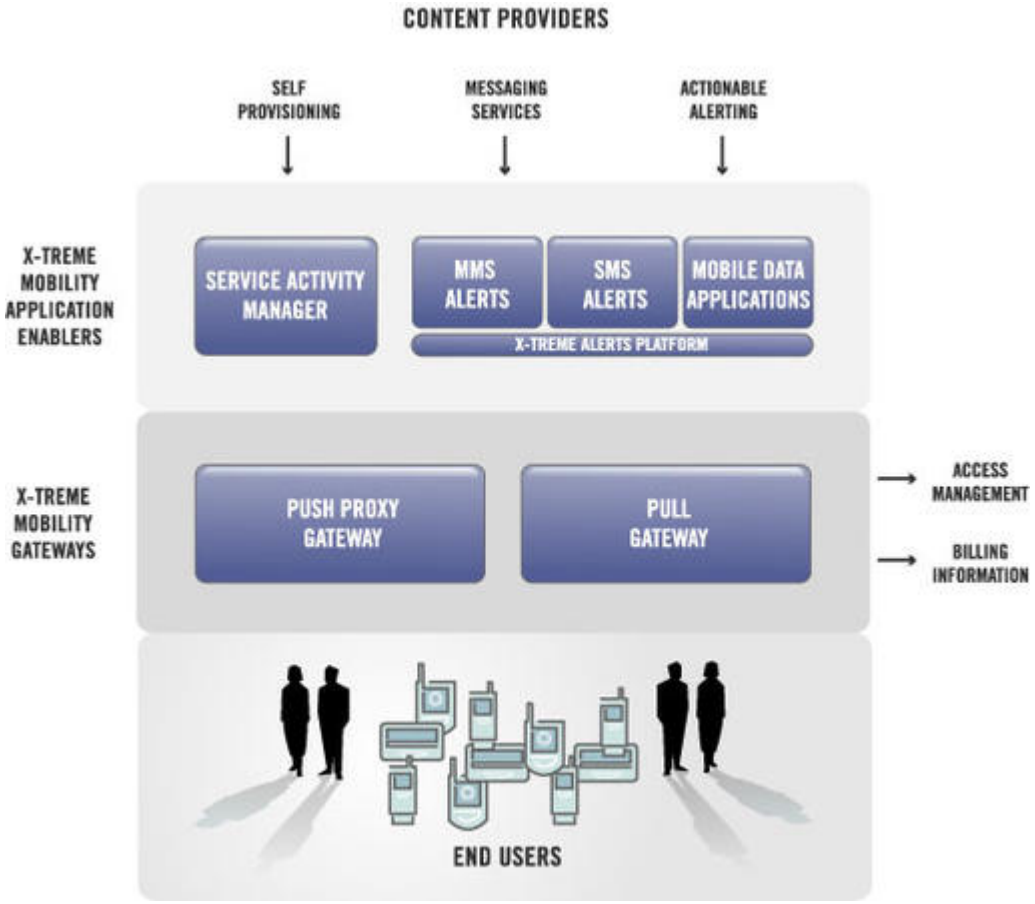
LynuxWorks now is offering a VxWorks Compatibility Layer package to help streamline the porting of VxWorks code to its LynxOS real-time operating system (RTOS). Through the use of this package, applications originally written for VxWorks' flat-memory model can be used with LynxOS, a multithreaded, POSIX-compliant RTOS. The VxWorks Compatibility Layer maintains separate name spaces under LynxOS by allowing multiple virtual VxWorks environments to run simultaneously when required. The VxWorks porting kit also provides recommendations for identifying certain types of code that may require special attention. An extensive list of supported VxWorks calls and limitations on their use also is provided.

LynuxWorks, 855 Embedded Way, San Jose, California 95138, 800-255-5969, www.lynuxworks.com.

X-treme Alerts Platform

724 Solutions, a provider of next-generation IP-based network and data services, is offering its X-treme Alerts Platform (XAP) on Linux. XAP is an actionable alerting platform tool that allows mobile operators to send easily personalized, permission-based Short Message Service (SMS) and/or Multimedia Message Service (MMS) alerts to subscribers. The latest version of XAP, available in both hosted and in-house deployments, improves performance and reduces TCO for mobile network operators. Additionally, XAP can be used to stimulate adoption of SMS/telephony voting applications by prompting the subscriber to cast a vote. XAP is part of 724 Solutions' X-treme Mobility Suite, which provides a next-generation data network solution for enabling differentiated, personalized premium data services.

724 Solutions, Inc., 4101 Yonge Street, Suite 702, Toronto, Ontario, Canada M2P 1N6, 416-226-2900, www.724.com.



LC2430 Debian Laptop

LinuxCertified, Inc., has announced the release of its first Debian Certified Laptop, the LC2430. The Debian model is the newest addition to the LC2000 laptop series, and it comes with preconfigured Debian GNU/Linux. Intended as a UNIX workstation replacement, LC2430 comes with a SXGA+ screen, up to a 3.06GHz Pentium 4 processor (with hyperthreading turned on), up to 2GB of RAM and up to an 80GB disk drive. It also features an ATI RADEON Mobility 9000 graphics card, 64MB of integrated VGA RAM, an Accelerated OpenGL card, a DVD and CD-R/W combination drive and built-in 10/100 networking. The methodology used to test the Debian GNU/Linux distribution on this laptop focuses on installation, configuration and operation; details can be found on the LinuxCertified Web site.

Linux Certified, Inc., 1072 South De Anza Boulevard, Suite A107-19, San Jose, California 95129, 877-800-6873, www.linuxcertified.com.

Archive Index Issue Table of Contents

Advanced search