

Advanced search

Linux Journal Issue #128/December 2004



Features

Making Movies with Kino by *Olexiy Tykhomyrov and Denys Tonkonog*
Make watching your family videos fun again with tightly edited scenes, titles and effects.

Open-Source Learning Management with Moodle by *Abhijeet Chavan and Shireen Pavri*

Want to teach a class on-line? Share notes, answer questions and give exams with this educator-proven free software.

Generating Music Notation in Real Time by *Kevin C. Baird*

This piece of music is never the same twice, as it re-writes itself in response to audience feedback.

Indepth

Beating Spam and Viruses with amavisd-new and Maia Mailguard by *Robert LeBlanc*

Here's a spam and virus filter that gives users a second chance to rescue important mail from the virtual trash.

Revision Control with Arch: Maintenance and Advanced Use by *Nick Moffitt*

Can you manage a software project and take your laptop away on a trip to hack? Yes—with these change-management skills.

Embedded

Automating Manufacturing Processes with Linux by *Craig Swanson and Ryan Walsh*

Running a factory with ISO 9001:2000 quality and just-in-time delivery means you have to collect a lot of data. Midwest Tool & Die keeps up using RTLinux and a PostgreSQL database.

Toolbox

At the Forge [Aggregating Syndication Feeds](#) by Reuven M. Lerner
Kernel Korner [Unionfs: Bringing Filesystems Together](#) by Charles P. Wright and Erez Zadok
Cooking with Linux [Lights...Camera...Action!](#) by Marcel Gagné
Paranoid Penguin [Adding Clam Antivirus to Your Postfix Server](#) by Mick Bauer

Columns

Linux for Suits [Unusual Suspects](#) by Doc Searls
EOF [gnuLinEx: Foundation for an Information Society](#) by Dario Rapisardi

Reviews

[Monarch ULB 64 2005 Custom Workstation](#) by Chris DiBona

Departments

[From the Editor](#)
[Letters](#)
[upFRONT](#)
[New Products](#)
[Best of Technical Support](#)

[Archive Index](#)

[Advanced search](#)

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

[Advanced search](#)

Making Movies with Kino

Olexiy Tykhomyrov

Denis Tonkonog

Issue #128, December 2004

Today's inexpensive camcorders can shoot great video. All you need to do a real video project is the right editing tools and, of course, practice.

Every camcorder owner would like to turn raw video into something real through editing. It does not matter how much the camcorder box brags of special effects and so forth, you still need proper editing tools. Fortunately, with today's powerful PCs, Linux and an application called Kino, you can put together your own little Hollywood studio.

Kino is nonlinear editor software. It can capture raw video, edit and re-organise it, and export the final results. With additional free, extra-cool plugins, Kino gives you all the necessary tools to make good movies.

Producing a movie usually includes making a recording with a camcorder. We don't cover this, but start with explaining how to copy a recorded raw video onto a PC with Kino. Modern camcorders communicate with PCs through the IEEE1394 interface that Apple calls FireWire; Sony prefers the name i.Link. After capturing, you can edit your raw video, adding titles and creating effects. You also may want to improve your movie by adding, mixing and replacing sounds. Kino supports all of these functions. When your movie is finished, you may want to make a DVD, so you can play it on a standalone DVD player, or even make an MPEG4. The quality of compressed movies is lower than the original raw files, so if you really want high quality and to use all the advantages of your digital camcorder, you also can copy the movie back to the DV tape with Kino and your camcorder.

Hardware Required

To send the video stream from a camcorder to a PC and back, you need an IEEE1394 interface on both sides, the PC and camcorder. Check your PC; many modern computers, including laptops, have this interface already built-in. If not, you can buy an IEEE1394 card separately from many vendors. You certainly need a digital video camcorder. Your camcorder probably is either Digital 8 (D8 for short) or MiniDV.

To connect the camcorder to the PC, you need a cable. Camcorders usually have four-pin female connectors, and PCs have six-pin connectors. Take a look at your computer (or IEEE1394 card) and buy a 4-4 or 4-6 cable if you do not have the correct cable.

Obviously, you need a PC equipped with a large capacity hard disk. You must have a lot of free space. For instance, to copy a 60-minute MiniDV tape to your computer, about 12Gb is necessary. Further work might need about 15Gb for edited frames and sounds, so 27Gb of free space is the absolute minimum for making a movie from one hour of raw video. Depending on what you want to do, you may need an extra 14Gb to export your results to a new .dv file. While capturing, the computer has to record about 3.5MB per second, so the hard disk should work as quickly as possible. Any modern Ultra-DMA drive works if configured properly.

A 1GHz processor and 128MB of memory is the absolute minimum suitable configuration for working with video. To work comfortably, you could use more memory and possibly a more powerful CPU.

Software Required

As far as software, you need Kino and the libraries it uses, but this is not everything. Kino provides only basic tools for editing and has few effects. Tim Shead and Dan Denedy have been developing timfx, a set of Kino plugins that adds extra effects. Another plugin, called dvtile, by Alejandro Sierra is needed for adding titles.

Because Kino requires additional programs and libraries, installing from source is not a simple task (see the on-line Resources section for Kino's home page). Kino is available as a package for several distributions, including Debian, SuSE and Fedora. Installing a package is much easier than building from source.

Kino development is progressing quickly. While writing this article, the latest version was 0.7.3. In Debian 3.0 (stable), the Kino version is 0.5.0, and Debian 3.1 (the developing branch) suggests the latest version is 0.7.3. SuSE 9.1

suggests using Kino 0.7.0. To help you keep the programs up to date we created all necessary packages. See Resources for links.

Moving Raw Video to a PC

Once the software is installed, connect your camcorder to the PC by IEEE1394, and run the command `kino` from any X terminal or, under KDE or GNOME, from the Alt-F2 dialog. The Kino opening window is similar to the one shown in Figure 1.

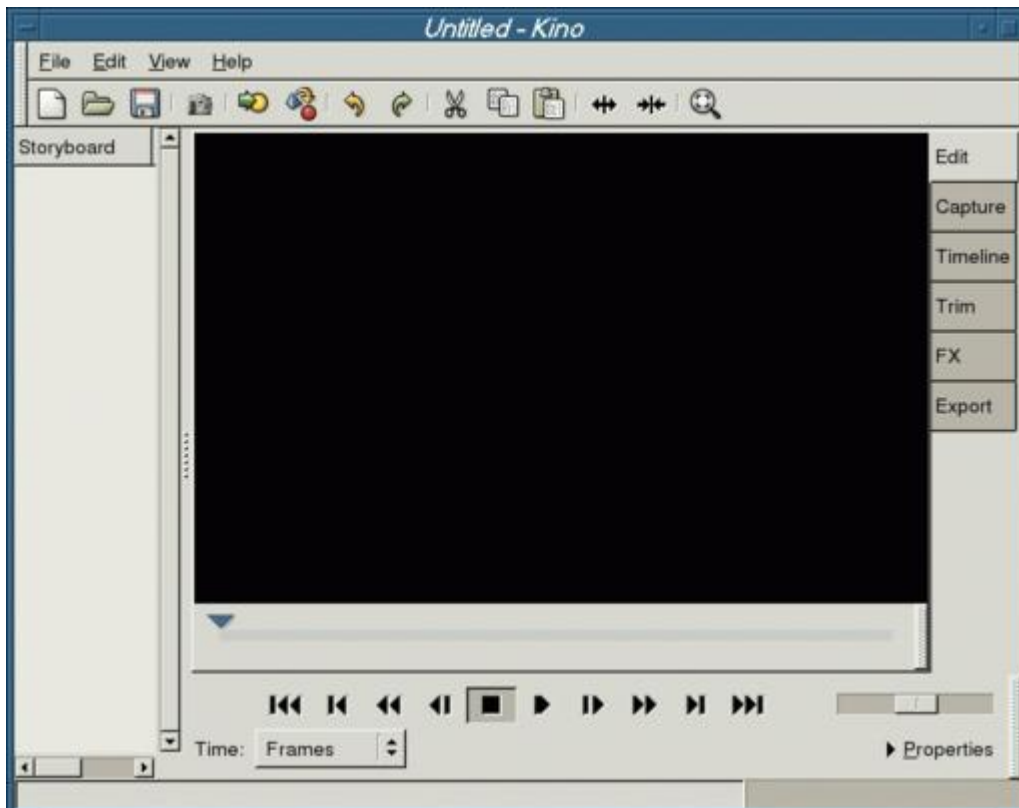


Figure 1. Kino's main window as seen when starting a new session.

The GUI is easy to understand. To capture video, choose the Capture tab on the right. Before starting to capture, however, it is best to check your preferences. Click on Edit→Preferences to see the default settings. Set Normalisation, Audio and Aspect Ratio according to your camcorder. Parameters depend on the camcorder you are using and the country where you bought it. The NTSC standard is used in the USA, Canada and Japan, and the PAL standard is used everywhere else. Camcorders usually have two audio modes, 16 bits and 14 bits, and the latter often is set by default. Change this to 16 bits before recording for better quality.

Captured frames can be stored in three formats, .dv and two kinds of .avi files. Generally, it does not matter which kind of format you use; we prefer to use raw DV.

After setting your preferences, press Capture, and enter a filename for the captured video without using an extension. You also can choose the Auto Split Files option so Kino saves each automatically recognised change of scene in a separate file, adding a number to the core filename. You can accept the default settings for all the rest.

Now, set your camcorder so it understands control signals. Usually this is in the Play mode, but check the manual if you are uncertain. Try controlling the camcorder with Kino by moving the tape back and forward. Try pressing the Play button in Kino. The video should start playing, and its output should be displayed in the main Kino window as well as on the camcorder screen. Notice the dropped frames field. Theoretically, this should be zero, but in practice, it's not bad if the first one or two frames are lost. If, however, the number of dropped frames keeps increasing, your hardware is too slow or misconfigured. If you cannot control the camcorder with Kino, try loading the raw1394 driver manually. As root, type `modprobe raw1394`.

If you have slow hardware, you also can try using `dvgrab`, a command-line tool for grabbing DV. This program is available from the Kino home page (see Resources). Before using it, exit your X session. Follow the directions from its man page. After grabbing the raw video, you can start Kino and load the captured files for editing as described below.

If the Play option operates correctly, you can start capturing. Go to 1–2 seconds before the position where you want to start, and click Capture. Kino starts the camcorder and captures. Pressing Stop at any time stops the capturing. Video transferred in this way appears in the scene list on the left part of the Kino screen. AutoSplit sometimes works incorrectly, but you can correct that later. Kino's window right after capture is shown in Figure 2.

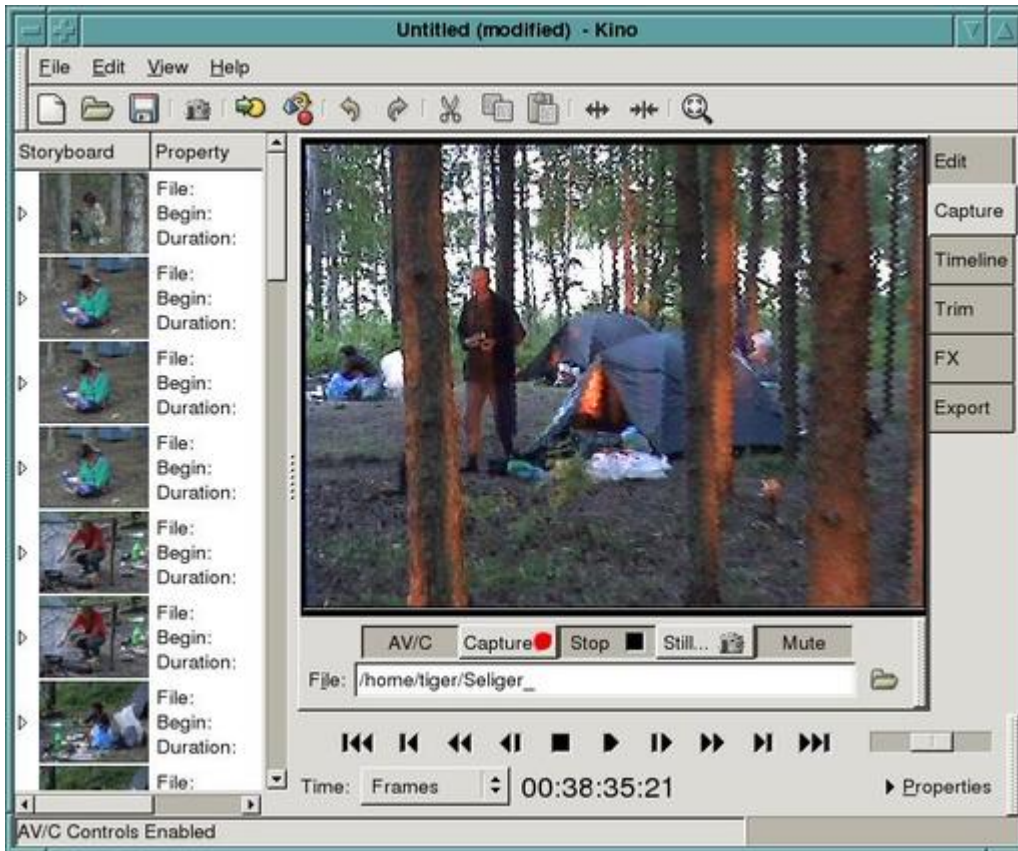


Figure 2. Kino's Window after Capturing

As soon as you finish capturing, select Save from the File menu, and save the project as a Synchronized Multimedia Integration Language (SMIL) file. An SMIL example is shown in Listing 1. Between the <seq> and </seq> tags, clip descriptions appear. Each one defines a simple or a complex scene. A simple scene is described by a <video...> command that points to the first and the last frames of the clip file that will be used in the movie; a complex scene is a set of simple scenes. The first frame of each scene is shown in the left panel of the Kino Storyboard.

Listing 1. An Example of an .smil Movie Project

```
<?xml version="1.0"?>
<smil >
  <seq>
    <video src="/mnt/RAW FILES/Paris001.dv" clipBegin="0" clipEnd="441"/>
  </seq>
  <seq>
    <video src="/mnt/RAW FILES/Paris002.dv" clipBegin="0" clipEnd="368"/>
  </seq>
  <seq>
    <video src="/mnt/RAW FILES/Paris003.dv" clipBegin="28" clipEnd="761"/>
    <video src="/mnt/RAW FILES/Paris004.dv" clipBegin="567" clipEnd="967"/>
    <video src="/mnt/RAW FILES/Paris008.dv" clipBegin="28" clipEnd="761"/>
  </seq>
  <seq>
    <video src="/mnt/RAW FILES/Paris004.dv" clipBegin="26" clipEnd="234"/>
  </seq>
</smil>
```

If your movie is assembled from many different tapes, put a new one in the camcorder and repeat the above steps.

In the storyboard near the first frame of a scene, you can find useful information, such as the name of the file where the scene is located, the time mark from which the scene starts and the duration of the scene.

While capturing, Kino displays the time according to the original source. In edit mode, timecode displays the running movie time. Time can be displayed in many formats. The simplest is the frame format. This is a frame counter that starts from zero. Other formats are much more usable by humans like seconds; minutes also are available. We like to use the industry-standard, Society of Motion Picture and Television Engineers (SMPTE) timecode format, which shows both the time and a frame counter, separated by semicolon. For instance, if you see the time noted as 00:07:40;15, it means 7 minutes, 40 seconds and 15 frames. You can select the time format you prefer from the Time pull-down menu.

Editing Video

Mozart is said not to have written drafts, but not everyone can be like Mozart. If you start recording too early or finish too late, a scene may need to be cut, or you may end up with bad frames, unnecessary frames or frames of the wrong duration. The duration of a scene usually is determined by the action. If there is no action, the scene duration should be no longer than 4–6 seconds. Shorter scenes look like flashes, and longer ones are boring.

To begin editing, look through the scenes. If you find only one or two problem frames in a scene, you either can remove them or split them with the next frame in the case of incorrect AutoSplit. To remove a scene, press the Edit tab on the right, point at the scene you want to delete in the storyboard and click on it. The main window shows the first frame of the scene. Click the scissor icon in the toolbar. The scene disappears from the storyboard, and the next scene's first frame is displayed in the main window.

Often you want to see an overview of the individual frames in a scene. With Kino, you can do this by clicking on Timeline. Figure 3 shows an example of the Kino Timeline.

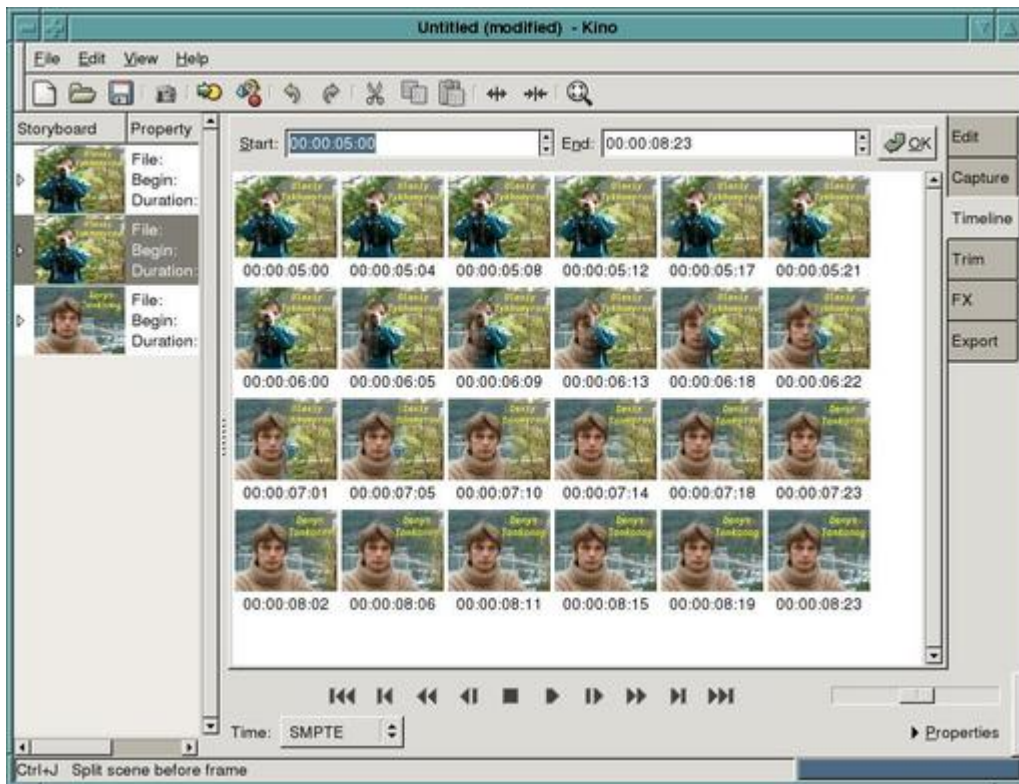


Figure 3. An Overview of the Scene with Timeline

If only a part needs to be cut from a scene, it should be split into bad and good parts. Use the transport control line to review and locate them. If you click on the split icon in the toolbar, the current frame starts a newly created scene. Be sure the current frame is in the scene you want to delete by clicking Cut. This method works well if you need to delete a central part of a scene. If you want to set the beginning or the end point of a scene accurately, it's better to use Trim.

Trim

Trim is a powerful tool for editing. The most basic use is to remedy problems with the first or last frames in a scene. To use Trim, select the scene in the Storyboard and click Trim. You should see a scrub bar and two text boxes; the one on the left is labeled In and the right one is labeled Out, as shown in Figure 4. These boxes show frame numbers for the starting and ending points for the current scene in the clip file. Use the control buttons or the triangle above the bar line to navigate to your new starting position. Use the single frame features for higher accuracy. Having specified the new starting position, click the left triangle below the line to change position. The In text box changes to reflect this selection. To trim the end of the scene, do the same thing with the ending point, and set it by clicking on the right triangle. If you are satisfied, click the red Apply button.



Figure 4. Editing with Trim

Exiting the Trim mode or choosing a new scene in the Storyboard to edit also applies any changes you made, so be careful while editing.

Under the In pointer is a text field indicating the current mode for trimming. Insert and Overwrite are the two trim modes and are similar to using a text editor. With Overwrite, the currently selected scene in the Storyboard is replaced by the newest one; Insert adds a new scene before or after the currently selected scene. The Insert mode has no Apply button, only Before and After icons. This means you must make your selection to apply the changes.

Put Scenes in Line

When all of your scenes or episodes are free of unwanted frames, put them in the right order. An easy way to reorder them is by using Storyboard. Click Edit first, then click a scene in the Storyboard you want to move and click Cut. Then, select the scene you want to follow it, and click Paste. The cut scene appears before the selected scene.

An even easier way to re-order scenes is by using drag and drop. In the Edit mode, click on the scene you want to move, hold down the mouse button and drag it to the required place in the Storyboard list. The scene appears at this position after you release the mouse button. While dragging and dropping, pay attention to the thin line that indicates the current target position. Drop does not work if the target shows a dotted rectangle. You also can insert previously

captured video clips into your current movie project. The Commands/Insert Movie button puts the selected file at the current position, and Commands/Append Movie follows it. Inserted files are split into scenes automatically if required.

Joining Scenes with Effects

After your scenes are assembled, the video part of your movie is almost done, but you still can add some of the effects that Kino and timfx provide. Be careful with effects, though—too many effects can distract viewers. You don't have to put an effect on every scene change, just like you don't have to use a new font in every section of a document. The simplest Kino effect is the Barn Door. Let's join two scenes with it.

Click FX→Video Transition. Select Barn Door Wipe from the pull-down menu. In the other field, choose what kind of door you want. Figure 5 shows a vertical one. The storyboard shows the first frame of the current scene that we want to join with the next one. So we select Frames following and Forward. To see the result before rendering, click Preview. Play with other options of this effect and preview them. When you are satisfied with a result, press Render.

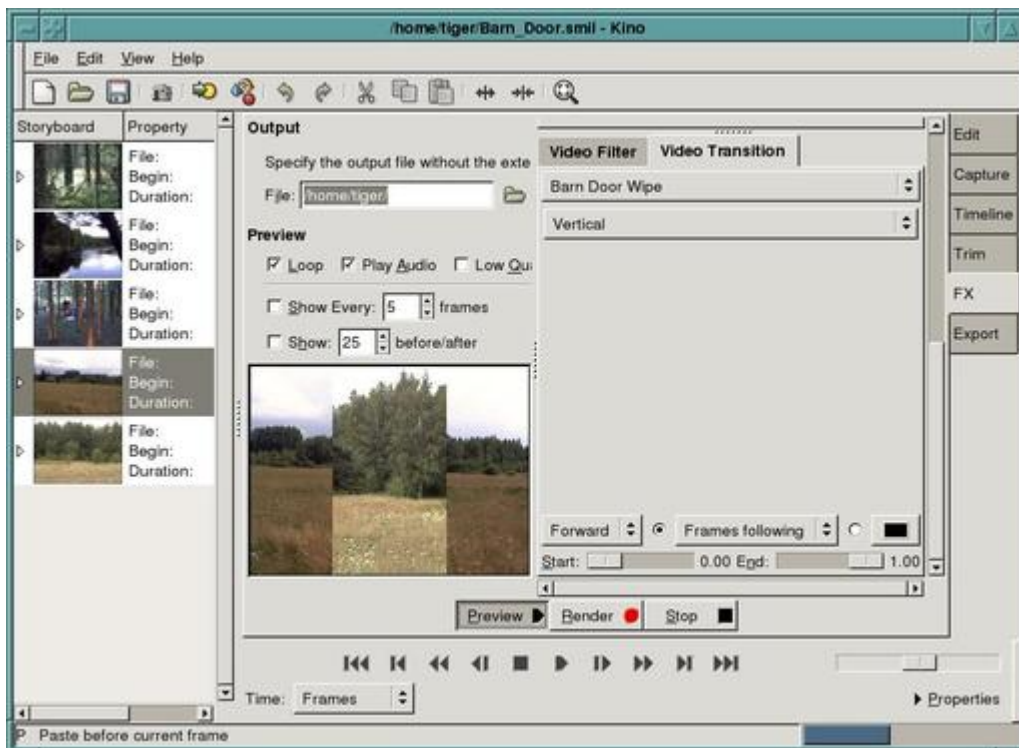


Figure 5. Making the Kino Barn Door Wipe effect—a preview of the effect is shown in the special window.

The render button has a red spot on it to show its importance. While rendering, Kino creates a new file. All previous editing keeps the original raw files untouched—only the SMIL has been changed. Now you can create a new file that is included into the project automatically.

Kino provides the Barn Door effect itself. Additional effects are available through timfx. Figure 6 shows the complete list. Many effects are well known, such as what we illustrate here with creating titles.

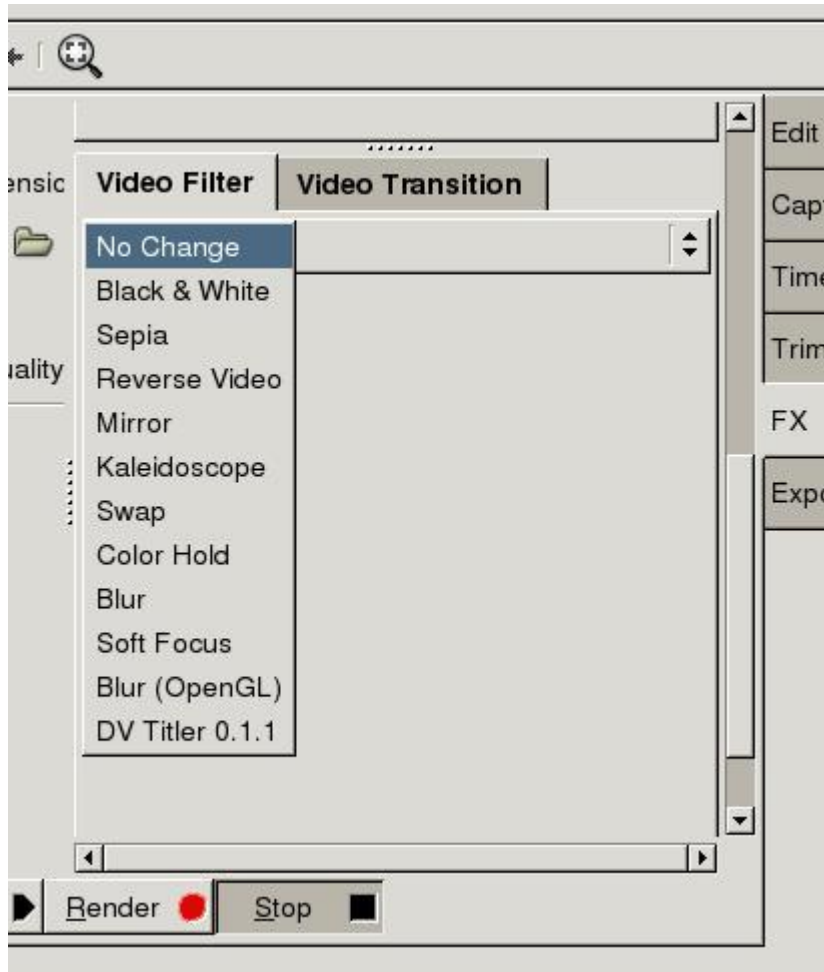


Figure 6. Video Filters Menu

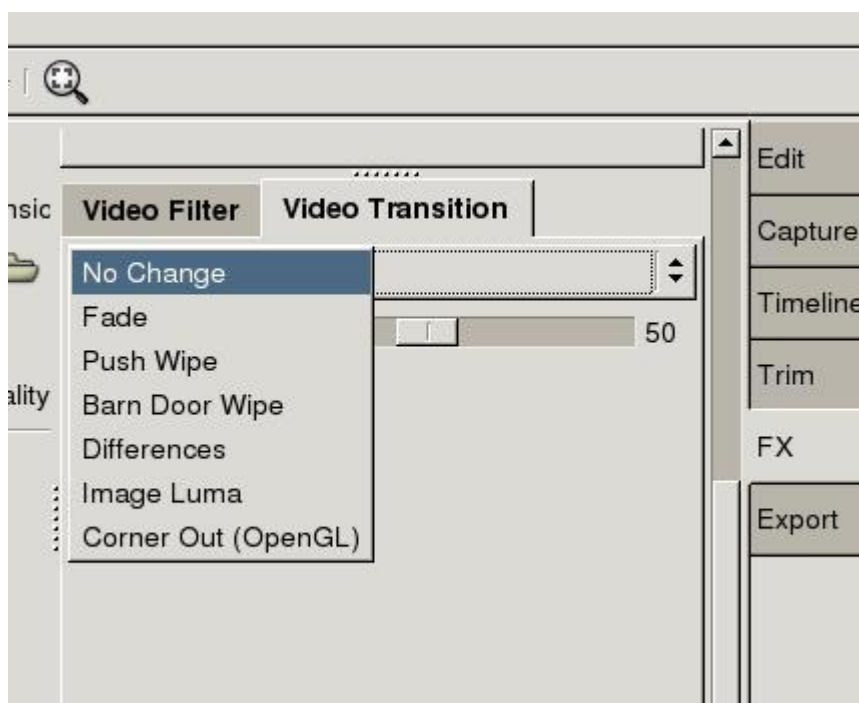


Figure 7. Video Transition Menu

Kino also can change the speed of your video. This function is used to add comic effects or make moments seem longer. To change the speed, use the scale bar under Advanced Options and push the Speed button. You also can activate Reverse to play your clip from the end back to the starting point. Changing the speed and reversing can be combined with the Transition or Filter effects.

Making Titles

If you want to add text comments to an episode, dvtitler can help. As an example, let's create a simple album that represents the characters acting in our movie. You even can create it using photos with an image editor such as The GIMP.

First, take a picture of the person and resize it for inserting into the movie. For PAL, the size should be 720×576, for NTSC 720×480.

Let's assume the following for this example. The first person will be shown for five seconds, then there will be four seconds for changing, and then five seconds again for the next actor.

From the FX menu, select Create $(5+4)\text{sec} \times 25 = 225$ frames. To keep the number of frames even, we choose Create From File → 226 frames, as shown in Figure 8. At the same time, we add the title "Denys Tonkonog". It consists of two centered lines and initially is located on the top right of the frame. As the final position is the same, the title does not move in the frames. The X offset and Y offset are used to keep the complete title in the frames, even if the movie is shown on TV, due to the possibly limited size of TV screens. The screenshot of the title preview is shown in Figure 9.



Figure 8. A Preview of Creating from File Frames

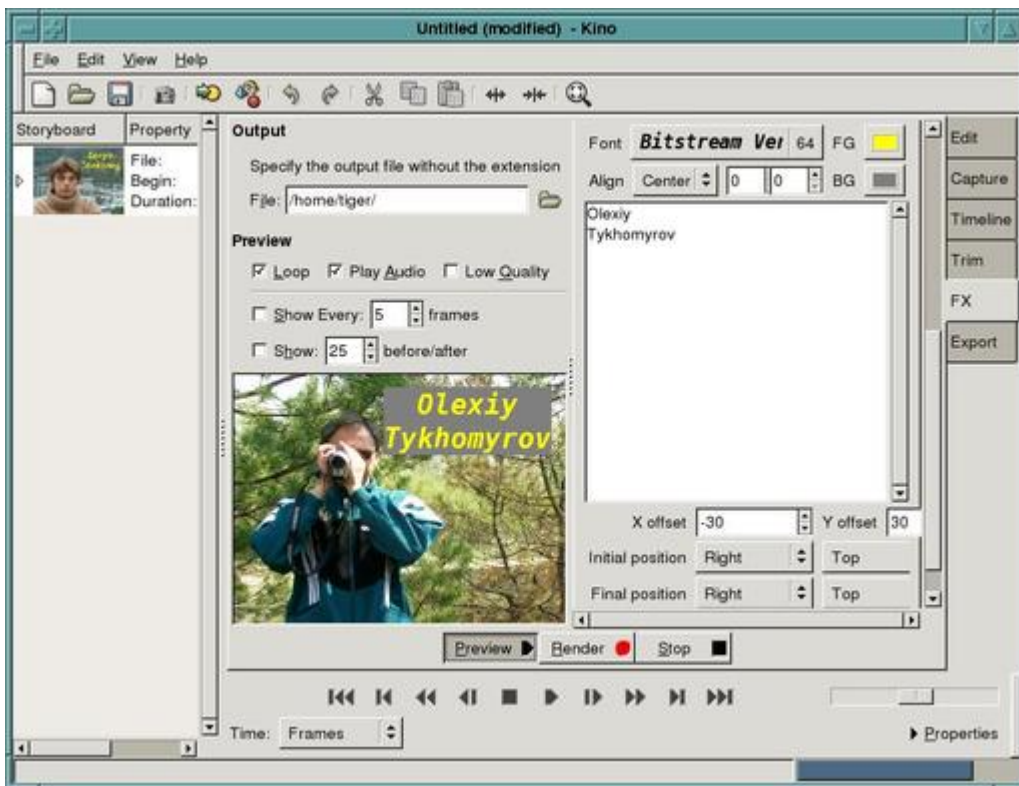


Figure 9. Making the Title "Olexiy Tykhomyrov"



Figure 10. A Preview of Joining Titles with Image Luma

We create another title, “Olexiy Tykhomyrov”, in the same manner. Then, we go back to Edit mode and use Split to separate the parts for applying the effect. Instead of creating two title scenes, we create four. They are five seconds in duration, then four seconds, four seconds again and then five seconds. The two central scenes (both four seconds) are joined into one with Image Luma.

In FX mode, click on the second scene under the Storyboard. Under Video Transition, select Image Luma. Next, specify the file with luma image. In the example (Figure 10), we used a standard gradient file left_to_right.png. Do not forget to change the mode from Insert to Overwrite. Once rendered, you have three scenes instead of four. The Timeline of the scene with changing titles is shown in Figure 3. Using Image Luma with different files can help you create many interesting effects.

Sound

With Kino, you can change the sound in your scenes completely, add sound to a scene, mix music or leave the soundtracks untouched. Finding suitable music for mixing or replacing might be the most difficult part of the work. For noncommercial projects, check Creative Commons.

Kino works with sounds in WAV format, not MP3. If you have an MP3 file and want to use it in your film, you have to convert it to WAV. We prefer to use mpg123:

```
mpg123 --wav foo.wav foo.mp3.
```

If you are using live sound, it's better to export soundtracks from a .dv source because sound is lost with cut video scenes. To edit sound, we use snd. Find the starting and ending points of the scene to use the sounds, then go to Audio. Select the parameters as needed, including From and To (Figure 11). Do not forget to specify the output file without an extension.

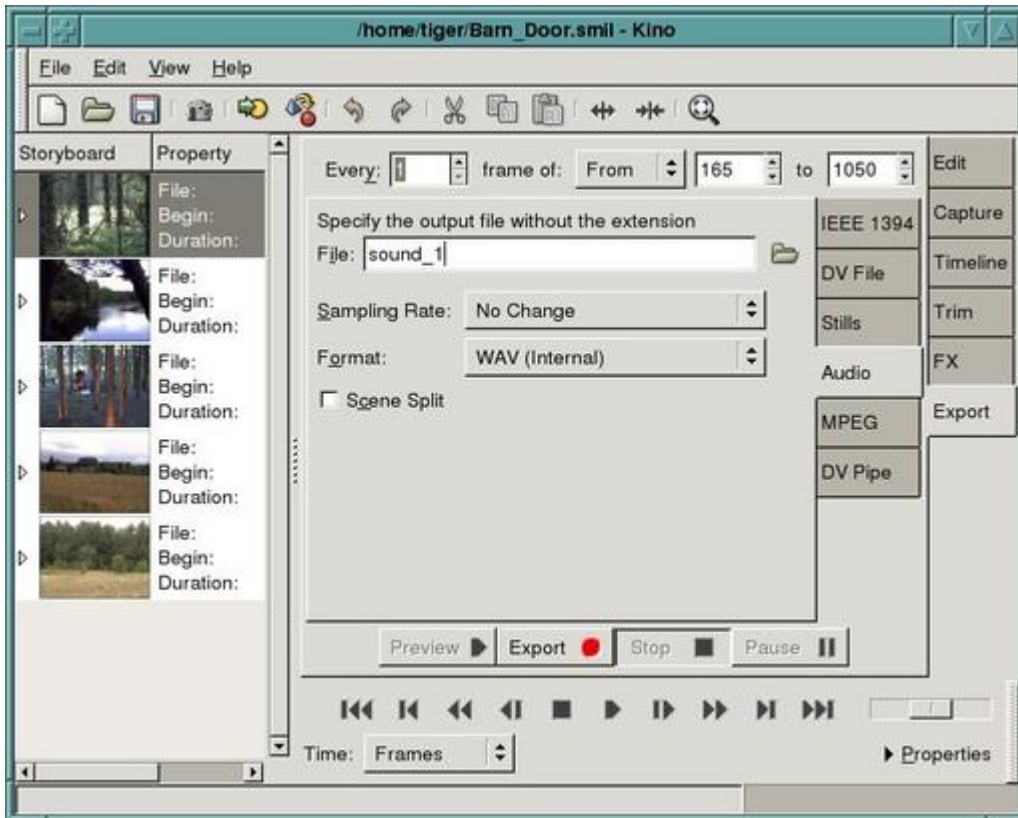


Figure 11. Saving Sounds from the Scene

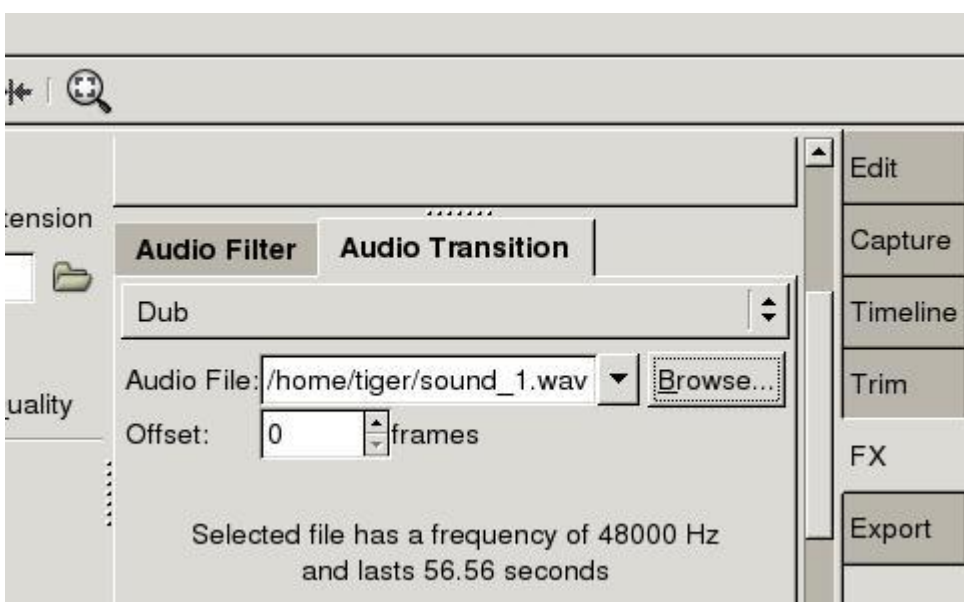


Figure 12. Audio Transition Menu

You can use Split to join several scenes into one in order to apply sounds. Be sure the scene for applying sounds is selected in the Storyboard. Next, click on FX and select Audio Transition. Choose Dub for applying sounds from the file or Mix for mixing the file with what already is in the scene. Add your audio file and preview the scene. Play with the parameters, and when you're satisfied, press Render. Rendering may take some time.

Exporting the Movie

After putting everything together, output the results. Although it is possible to create many output formats, to keep the original quality, we recommend that you output your film back to the tape to save hard disk space. Do this by selecting the Export tab and then IEEE1394 from the menu. Before exporting through this interface, set up the dv1394 device and driver. First, issue one of the two following commands, depending on your camcorder. The command creates the dv1394 device.

For PAL systems the command is:

```
mknod -m 666 /dev/dv1394 c 171 34
```

For NTSC systems the command is different:

```
mknod -m 666 /dev/dv1394 c 171 32
```

Now, still as root, load the driver with the command:

```
modprobe dv1394
```

Next, as an ordinary user from Kino, select Preferences→IEEE1394. A window similar to Figure 13 appears in the screen. If you captured with Kino, the part titled DV Capture is filled in correctly; your camcorder is recognised as a VCR (AV/C) Control device. The camcorder should be turned on so it understands control signals (usually Play mode).

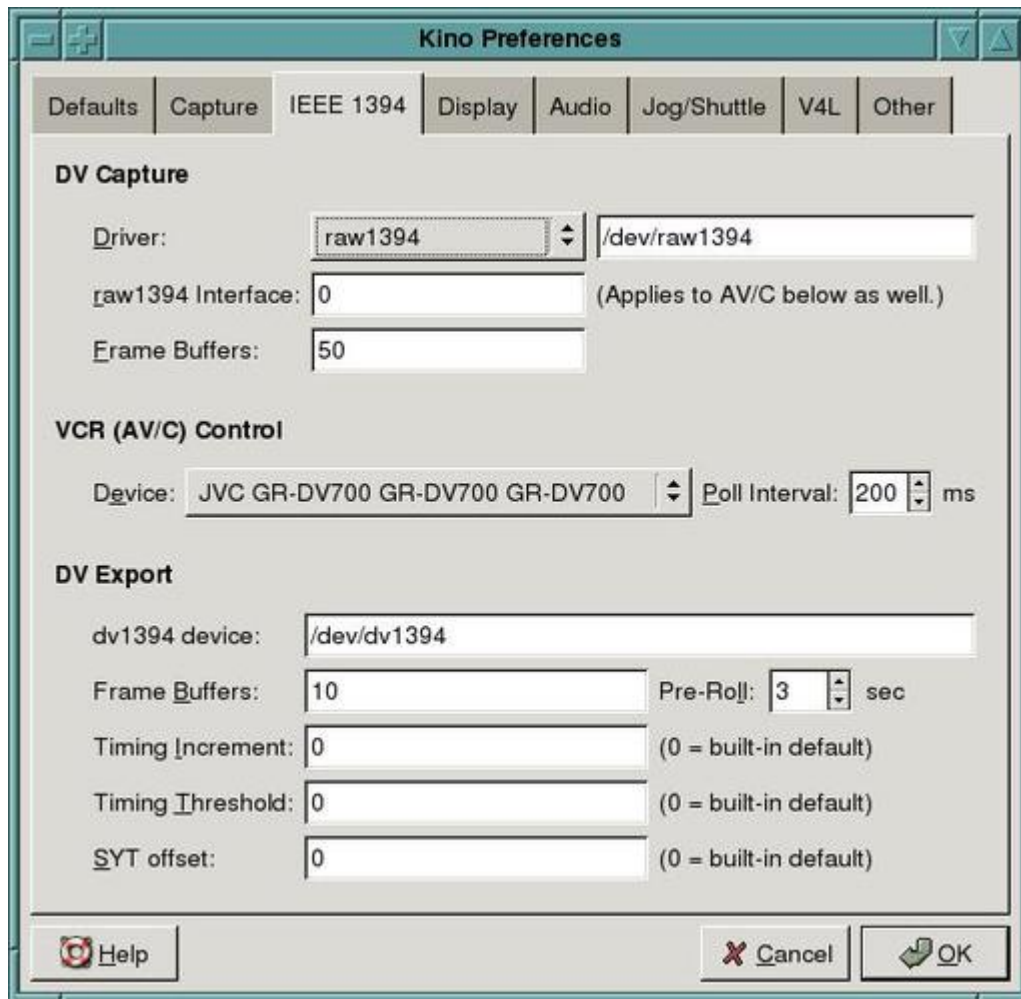


Figure 13. Setting Kino for Exporting Film with IEEE1394 to the Camcorder

To output through the IEEE1394 interface, change the clue line dv1394 device to /dev/dv1394. Then click OK to close the setup menu. Next, click Export and select IEEE1394. You should see something like Figure 14.



Figure 14. Exporting the Movie Back to the Camcorder

Notice the information line on the bottom of Figure 14. Here, Kino informs you about dv1394 device availability. If it complains, wait about 20–40 seconds. If the error repeats after this interval, read more information about the device on the Linux1394 home page.

Next, select Preview. The movie should start playing on the camcorder screen. Now, put the tape in the camcorder to the position where you want to write, and select the Export button to start writing the movie to the tape. Check what your camcorder indicates according to its User's Manual. Usually it shows a line like DV Input and possibly provides additional DV export information.

If everything is okay, you can stop the camcorder, put the tape exactly at the position from where you want to write the movie and start exporting. The time needed for writing is equal to the movie duration. While exporting, Kino indicates the elapsed and remaining time in the bottom of the export window.

You also can export the movie as a file or a set of files. Kino, with additional programs, provides the following:

- A DV file that stores the movie in a new file format, which can be used further for compression and exporting without the Kino interface.
- Stills save the movie as a sequence of images in JPEG format.
- MPEG codes the movie to MPEG or DivX.

- Audio lets you save only the audio tracks.
- DV pipe gives you a nice tool to create your own method of saving movies. It also includes standard features, such as exporting to MPEG1 for Video CD or to MPEG2.

Remember that all export features might not work on the box you are currently using; installing extra programs might be necessary. Exporting to a DV file should work and reflect the general rules similar to saving a movie on the tape with the camcorder and IEEE1394 interface.

To export the movie into a single .dv file, select Export/DV File. Disable the option Auto Split Files, and specify All as Export Range to export the whole movie from the scene list. Select Type File—we prefer Raw DV. Put zero values into the fields Frame per File and Max File name, as shown in Figure 15. Click Export. Kino starts exporting, indicating the estimated time necessary to finish the process.



Figure 15. Exporting Movie in a .dv File in Progress

Some Final Remarks

Kino still is under development and may crash sporadically when you are using Edit or Trim, so make sure you save your movie project regularly. Although Kino watches the files it creates, it may lose control of them in the case of a crash. In order to keep your hard disk clean, do not forget remove unused .dv files.

Acknowledgements

The authors sincerely thank Professor Paul Bartholdi, Observatoire de Geneve, the International Centre for Theoretical Physics, Italy, for providing a real Internet connection that is impossible from Dnepropetrovsk National University.

Resources for this article: www.linuxjournal.com/article/7816.

Olexiy Tykhomyrov (tiger@ff.dsu.dp.ua) has been using Linux since 1994. He works for the Department of Experimental Physics at Dnepropetrovsk National University and teaches physics and communications. He loves his son Misha who calls him Tiger because some of his students are afraid him. Tiger likes swimming and traveling.

Denis Tonkonog, a former student of Tiger, also works at Dnepropetrovsk National University and likes traveling and fishing with a gun. Friends call him Black Cat but nobody explains why. He can be reached by e-mail at denis@ff.dsu.dp.ua.

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

[Advanced search](#)

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

[Advanced search](#)

Open-Source Learning Management with Moodle

Abhijeet Chavan

Shireen Pavri

Issue #128, December 2004

Combining the features of a content management system, bulletin board, and on-line grade book, Moodle meets a growing demand for on-line education.

Currently, an explosion is occurring in the demand for distance education in the US. Large numbers of high school graduates are going on to college, and more adults are pursuing a college education. The demographics of college students also are changing, with more students juggling work and family responsibilities than ever before, which necessitates easier access to education. Furthermore, the changes in knowledge and skills catalyze the need for ongoing professional development of the existing work force. In response, corporations increasingly are turning to distance education options for their employees.

A learning management system (LMS) is a software system used to deliver on-line education. Alternate terms often used are managed learning environment, virtual learning environment, course management system or learning support system. Today, most LMSes make extensive use of the Web and include features such as discussion forums, chats, journals, automated testing and grading tools and student tracking. LMSes also are used to supplement regular face-to-face courses. They are used in universities, schools and by businesses to deliver corporate training.

Start up and maintenance costs for on-line education typically have been high, with proprietary software solutions such as Blackboard and WebCT being the dominant choice amongst academic institutions and corporations. But cost is not the only or even the prime reason to look beyond available proprietary LMS solutions. The ability to modify software is an important consideration for many institutions that need to address specific teaching and learning requirements. Others need to integrate a new LMS with existing systems.

Several open-source projects have emerged to meet the growing interest in open-source LMSes (see the on-line Resources). In this article, we look at one popular open-source LMS, Moodle.

Introducing Moodle

In classic open-source fashion, Moodle was born out of a need to scratch an itch. Frustrated by proprietary alternatives, Martin Dougaimas, then a PhD candidate in Education with a background in computer science, started Moodle in 1999. Version 1.0 was released in August 2002. Since then, Moodle has continued to evolve at a rapid rate, managed by Martin in Australia and propelled by an active world-wide community of users and developers.

A single Moodle Web site can host a large number of courses. Each course is managed by one or more teachers. Courses can contain activities such as discussion forums, student journals, quizzes, surveys, assignments, chats and workshops. Moodle includes support for grading, file uploads, user logging and tracking, multimedia, e-mail integration and many other features, all comparable to those available in proprietary LMSes.

Moodle is developed on the popular LAMP platform—GNU/Linux, Apache, MySQL and PHP. Part of Moodle's attraction is it can run on almost any server that can run PHP. In addition, PostgreSQL can be used instead of MySQL. The flexible technical requirements make it possible to install and evaluate Moodle on almost any computer and even run it on shared Web servers managed by Web hosting providers. Moodle is offered under the GNU General Public License. The GPL, well-documented PHP code, an active developer community and a modular design make it possible to customize Moodle and integrate it with other open-source software. For users, all Moodle requires is a Web browser and an Internet connection.

Most LMSes are instructor-oriented and largely concerned with how course content is delivered. Moodle is based on a learner-oriented philosophy called social constructionist pedagogy, in which students are involved in constructing their own knowledge. The concepts behind this philosophy of learning are that learners actively construct new knowledge by tinkering, and they learn more by explaining what they have learned to others and by adopting a more subjective stance to the knowledge being created. These ideas run parallel to the way open-source development works, in which the developers also often are users, everyone is free to tinker with the software and code is constructed, peer-reviewed and refined by the means of an open discussion. This philosophy is the basis for the unusual name of this project. The Moodle Web site explains the origin of the name:

The word Moodle was originally an acronym for Modular Object-Oriented Dynamic Learning Environment....It's also a verb that describes the process of lazily meandering through something, doing things as it occurs to you to do them, an enjoyable tinkering that often leads to insight and creativity.

The social construction pedagogy is reflected in the design and choice of Moodle features. For example, one of Moodle's features is every course can have a glossary of terms. The glossary can be set up to allow course participants to add their own terms and definitions. Taking it a step further, Moodle allows comments to be attached to each term, enabling participants to refine and clarify these definitions.

Installation and Configuration

At the time of this writing, the latest stable release of Moodle is version 1.3.1, which was released on June 5, 2004. If you are interested in experimenting with the newest features, you also can download the nightly development packages. Both the stable release and the development versions are available from anonymous CVS. In our experience, we have been able to upgrade Moodle installations by way of CVS without problems. The stable CVS branch is new in v.1.3 and promises a convenient way to maintain a Moodle installation.

Installing Moodle on a LAMP system is straightforward and well-documented. After unpacking the downloaded package, place all files and folders into your Web server's documents directory. Create a MySQL database and account. Moodle needs a separate data directory to store some files, such as user-uploaded images. This directory should not be accessible directly over the Web. You can protect it either by using an .htaccess file or by placing the directory outside the Web server's documents directory.

The default Apache and PHP settings on most Web servers should be adequate. PHP sessions support and file uploading need to be enabled. Also, PHP safe mode needs to be disabled. Some Web hosting providers do not allow disabling PHP safe mode while others do, so check in advance.

A single file, config.php, stores the basic configuration settings, such as database information, Web site URL, directory paths and permissions. Make a copy of the config-dist.php file provided by Moodle, name it config.php and edit it using your favorite text editor. This thoughtful arrangement is useful when you upgrade Moodle. Moodle's config-dist.php is upgraded, but your config.php, which contains settings specific to your installation, is left untouched. The config.php contains detailed instructions and examples.

Next, visit the main page of your Moodle Web site with a Web browser. From this point on, Moodle handles its own installation over the Web, setting up the database and creating tables. The defaults should work to get you started, and you always can customize them later. Finally, you are asked to create an administrator user account. Successful creation of the user account completes the Moodle installation, and you are returned to the home page of your new Moodle site.

Once Moodle is installed, almost all regular administrative activities can be carried out by using a Web browser. When logged in as the administrator, a block containing administration links appears in the left column of the main page after installation (Figure 1). The Configuration link in this block opens up a control panel that allows the administrator to control all aspects of the Moodle site using a Web browser. Again, every setting is meticulously documented and examples are provided.



Figure 1. Moodle Main Page after Installation

The Variables panel controls the basic operation of the Moodle site. In most cases, the defaults should work fine. The Site Settings panel is where you set the name of the Web site. This also is where you can change the words used to refer to teachers and students—another example of Moodle's flexibility. For example, you can specify that teachers should be referred to as moderators or facilitators and students should be referred to as participants. At this point, the Moodle site is ready. You can start creating courses and adding users.

Administration

To add a new course, while logged in as the administrator, follow the Courses link in the Administrator block on the main page of the Moodle site and choose

the button to add a new course. Courses are classified into categories and each course has to belong to a category. Miscellaneous is the default category. You can add, delete or hide categories as needed.

Moodle provides three course formats: weekly, topics and social. The weekly format is suitable for courses organized into weekly activities. The topics format is suitable for courses organized into topics instead of weeks. The social format is organized around a single discussion forum. Choose a format and create the course (Figure 2). Once a course has been created, the assigned teacher for that course can modify course settings at any time.

Moodle Demo Logout

Demo -> Administration -> Course categories -> Add a new course

Edit course settings

Category: Miscellaneous ?

Full name: ?

Short name: ?

Summary:

Arial 1 (8 pt) Heading 1 **B** *I* U ~~S~~ x_2 x^2

Write a concise and interesting paragraph here that explains what this course is about

Path: body ?

Format: Topics format ?

Course start 15 September 2004 ?

Figure 2. Creating a New Course

After you create the course, you are taken to the main page of the newly created course. Choose the Turn editing on button at the top right corner of the page. With the editing turned on, tiny icons appear all over the page. These icons allow you to reposition blocks of content on the page as well as add, edit or delete resources and activities in the course (See Figure 3).



Figure 3. Adding Course Materials

The Users link in the administrator block allows the administrator to add users to the Moodle site. Moodle has convenient user management features. New users can create user accounts themselves by providing an e-mail address. Moodle handles the signup process by confirming the e-mail address, creating the account and generating a password. A user that has forgotten the password can request to have it sent to his or her e-mail address. The administrator also can import multiple users from an external comma-delimited file. The administrator can assign teachers for a course and enroll students to that course.

Customization

Moodle comes with 15 themes that change the look of a Moodle Web site, and you can switch themes directly from the Configuration panel. It's easy to create new themes, too. Each theme is contained in a subdirectory of the main theme directory. To create a custom theme, copy one of the existing theme folders and give it a new name. The folder for each theme contains certain standard files, such as config.php and styles.php. The easiest way to change the look of the theme is to tweak the file styles.php, which modifies the Cascading Style Sheet (CSS) used for that theme. Want to add a logo at the top of the page? Customize header.html. Any new subdirectory in the main theme folder automatically appears in the Configuration panel for themes. No special installation step is required.

Moodle's world-wide popularity—1,900 registered sites from 90 countries at last count—could be attributed to the fact that it is available in 40 different languages, from Afrikaans to Turkish. Similar to the plugin arrangement for themes, each language pack is a subdirectory in the lang directory. Switching

languages is done easily through the Web control panel for languages. But, Moodle takes flexibility a step further by making it possible for an administrator to change all the phrases used in a language pack. For example, if you want to change the word “courses” to “classes” throughout the Moodle site, it can be changed once in the language pack. The terms in the language pack can be edited directly from the Web-based configuration panel.

The standard package of Moodle comes with over 15 modules for various activities, such as discussion forums, chats, assignments, journals, quizzes and surveys. Additional optional modules are available on the Moodle Web site. Installing a module usually involves copying the module's files into a subdirectory under the mod directory. Some modules come with related language files that need to be copied over to the appropriate subdirectory under the lang directory, depending on which language pack you are using. Using the configuration panel, it is possible to delete a module or change its default settings.

The Web control panel also allows the administrator to set up automated backups. Options control what is archived in the backup file and when. For example, you can omit large log files from being included in the backup. Teachers can make backups of courses, too. A backup without user data can be used to set up a course for the next semester. The backup file is in XML format, meaning your data is never locked in a proprietary format and you maintain control over your own data.

Why Moodle?

The Moodle Developers Manual lays out the development goals of the project:

- Moodle should run on the widest variety of platforms.
- Moodle should be easy to install, learn and modify.
- It should be easy to upgrade from one version to the next.
- It should be modular to allow for growth.
- It should be able to be used in conjunction with other systems.

In our experience, the Moodle project has managed to stay true to these objectives even as it evolves rapidly. The quick pace of development is apparent in the frequent releases. For example, Moodle users had expressed interest in an integrated calendar, a feature that Moodle lacked. In January 2004, Moodle users and developers started discussing what calendaring features were needed and how they should be implemented. A few months later, the May 2004 release included an integrated calendar.

An often-repeated criticism of open-source software is it is only for IT experts and is too difficult for basic users to install and use. Yet, over 66% of Moodle users who responded to a Moodle survey identified themselves as teachers, on-line learning researchers or educational administrators.

Managing an LMS can be a complex task. What we like about Moodle is it does not hide this complexity. Its detailed on-line help, examples and sensible defaults assist the user in installing, administering and using the LMS. The greatest strength of Moodle is the community that has grown around the project. Both developers and users participate in Moodle's active discussion forums, sharing tips, posting code snippets, helping new users, sharing resources and debating new ideas. Moodle's low cost, flexibility and ease of use helps bring LMS technology within the reach of those with limited technical and financial resources. Moodle is a fine example of how and why open source works.

Resources for this article: www.linuxjournal.com/article/7817.

Abhijeet Chavan is the Chief Technology Officer of Urban Insight, Inc. He also is the co-founder and co-editor of *Planetizen* (www.planetizen.com).

Dr Shireen Pavri is an Associate Professor in the College of Education at California State University, Long Beach.

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

[Advanced search](#)

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

[Advanced search](#)

Generating Music Notation in Real Time

Kevin C. Baird

Issue #128, December 2004

Kevin Baird's *No Clergy*, his doctoral dissertation piece in music composition, allows audience members to influence the musical notation presented on-screen to musicians in a live interactive performance.

Composers from a variety of backgrounds have been interested in having greater freedom in musical performances. This article describes another attempt in this tradition.

In his *Klavierstcke*, Karlheinz Stockhausen allowed performers to order prewritten material as they saw fit. Earle Brown allowed conductors to mix and match material in his *Available Forms* pieces, and in his graphic scores, he allowed performers essentially to improvise whatever the images in the score inspired them to play. Improvisation obviously is critical to jazz, as well.

In all of these examples, the composer gives this increased freedom only to performers or the conductor, not to the audience. My doctoral dissertation piece *No Clergy* and its associated programs constitute one attempt to give similar freedoms to the audience.

In *No Clergy*, performers play on-screen notation presented through a Web browser. The audience has Web browsers open to pages with standard CGI forms, allowing them to react to what they're hearing by entering data. The piece's scripts then process the data, affecting the subsequent pages of notation presented to the performers.

No Clergy has specific technical, logistical and esthetic requirements. The processing must occur in real time. The audience's and performers' interfaces for the piece must be familiar and comfortable. The piece must be transportable—able to be performed in any location with minimal setup requirements.

The user, who serves as the “conductor” and is either me or someone filling a similar role in a performance where I am absent, starts the piece by running the bash script `setup.sh`. The script generates, with Python, a markup file for GNU Lilypond, a music typesetting program described in greater detail later in the article. It then processes that Lilypond file into a PNG image, as shown in Listing 1. Portions of the script that are not shown also perform cleanup actions to remove old data and ending actions to place images in the appropriate Web directories.

Listing 1. Excerpts from `setup.sh`

```
python -o NoClergy/Python/make_ly.py \  
clar > lilypond/ly/clar.ly  
lilypond --png -o lilypond/out/ lilypond/ly/clar.ly
```

The image displays a musical score for a piece titled "No Clergy" by Kevin C. Baird. The score is for a Clarinet and is set in 3/4 time. It begins with a tempo marking of quarter note = 233. The music is written on a single staff in treble clef. The score includes various dynamic markings such as *mf*, *pp*, *ppp*, *ff*, and *mf*. There are also articulation marks like accents and slurs. The piece is marked with a first ending bracket. The score is presented in a clean, professional layout with a white background and black notation.

Figure 1. Sample Initial Output for a Clarinet

Later, when user data are available, a similar bash script called `noclergy.sh` (Listing 2) reads the previous data and generates subsequent pages of notation. The script `mutate_config.py` reads the user data and updates a config file, while `mutate.py` applies those changes to the musical material, performing actions similar to the `setup.sh` script described above, but this time with a previous example of output from which to work.

Listing 2. Excerpts from noclergy.sh

```
python -O NoClergy/Python/mutate_config.py
python -O NoClergy/Python/mutate.py 'lilypond/ly/'
```

I chose an object-oriented paradigm because the musical material contains multiple instances of similar types of data. Initially, the top-level Class was Score, defined as one page of notation for one instrument. Each Score has a transposition level appropriate for its instrument and can contain an arbitrary number of Measures. I chose 20 as good number of Measures to fit comfortably on a single page with reasonable legibility, and as a good duration of musical time between each updated page of notation.

For Non-Musicians

Transposition

Transposition is a musical term for changing the pitch of a piece of music. Some instruments sound lower or higher than normal and, therefore, require music written for them to be transposed. This practice allows performers to learn fingerings for an instrument family, rather than only one specific instrument.

Pitch Class

A note's pitch class is its pitch mod 12. Middle C is a pitch, whereas C is a pitch class. When musicians talk about a C or a B flat, they are referring to pitch class. A doubling of frequency is a rise of one octave, and differences in octave are what separate notes with different pitches but the same pitch class.

Each Measure has a meter, which determines how long it is and its internal rhythmic organization. It contains however many Notes fill that meter. The total number of Notes also varies based on each Note's duration.

A Note is defined as an individual sound or silence event within a Measure. It has a string called pitch, which is either r for a rest or an indication of its pitch class, such as c, cs for C sharp, d or ef for E flat. It also has an integer called octave, which is meaningful only for non-rests.

Notes also have durations, dynamics, which are variations in amplitude or volume, and articulations, which determine whether the note is accented, detached or sustained into the next note, among other things. Notes also can be tuplets, which are a particular type of rhythmic organization.

Tuplets are notes or rests that occur at a different rate than their note type would indicate. By far, the most common type of tuplet is the triplet. A set of three triplets occupy the same duration as two notes, so three 8th note triplets occur in the span of two normal 8th notes, or one beat in most instances. Quintuplets are five notes in the span of four, septuplets are seven in the span of four and so on. Unless otherwise indicated, a set of tuplets squeezes x notes in the span of y , where y is the highest power of 2 lower than x . Composers generally indicate deviations from this practice with $x:y$ notation on the tuplet set, so a 7:8 tuplet set stretches seven notes across the span of eight.

No Clergy uses a non-unique list of tuplet types—the number indicated by x in the preceding paragraph. This allows me to weight in favor of triplets and quintuplets, as is common. Anyone who wants to use my program to sound like Frank Zappa or Brian Ferneyhough can alter the tuplet list as they wish, although musicians should know that this initial version of the program does not yet support nested tuplets. Non-musician programmers probably can discern from the name that nested tuplets are sets of tuplets that themselves contain one or more sets of tuplets. The program also does not yet support dotted notes, which use a notation convention to show that a note should last half again as long.

When the scripts generate notation, they read a configuration file, updated by the `mutate_config.py` script mentioned above. The various `pc` variables shown in Listing 3 represent the percentage chance that a given note will have the characteristic in question, such as being a tuplet, being a rest as opposed to a note that sounds, having an explicit dynamic mark and having an explicit articulatory mark.

Listing 3. Configuration Variables

```
# No Clergy config.txt, written by script
tupletpc = 50
restpc = 25
dynpc = 25
artpc = 25
number_of_measures = 20
```

The scripts then generate a full Score's worth of musical material, constrained by the variables above. At this point, all of these data still exist only as Lists of Objects within a Python script. I chose MusicXML as the format for external storage.

MusicXML is a subset of XML specifically geared toward musical data. Developed by Recordare LLC, it is largely geared toward being an interchange format for music notation programs. It is also useful, however, as a generic storage format for musical data, as befits a dialect of XML.

Listing 4. Sample MusicXML Fragment for One Note

```
<note>
  <pitch>
    <step>b</step>
    <alter>-1</alter>
    <octave>3</octave>
  </pitch>
  <duration>16</duration>
  <type>sixteenth</type>
</note>
```

Musicians probably can figure out that this XML fragment represents a 16th note B flat in the third octave with no dynamic, articulatory or other special alterations.

The script stores the most recent MusicXML file for each instrument using the path convention `inst/yyyy_mo_dd-hh_mi_ss.xml`, where `inst` is the instrument name and other letter codes are time units. Upon reading an XML file, the script then moves it into a backup directory and compresses it with `bzip2`. Storage of old data is useful for documentation of specific performances, learning how audiences react to their role in the piece and debugging. Once the script has read in the XML data, it's ready to output for processing by Lilypond.

GNU Lilypond is a Scheme-based music typesetting program that uses a TeX-like backslash notation for formatting commands and has a particular focus on high-quality music engraving inspired by the best traditional hand engraving. It outputs to several high-resolution formats, including PostScript, DVI and PNG.

Listing 5. Sample of Lilypond Markup

```
| % MEASURE 2
\time 7/8 ef''8-\pp a'4 d'2-\marcato
```

In Listing 5, we see Lilypond markup for one measure of music. The vertical pipe represents a barline, the `% MEASURE 2` is a comment, the `\time` indicates the 7/8 meter, the `ef` makes the first note an E flat, the `''` raises the note two octaves higher than the baseline used by the program, the `8` gives the note a rhythmic value of an 8th note and the `-\pp` attaches a pianissimo dynamic indicator to the note. Two other notes follow. The measure is rendered as shown in Figure 2.



Figure 2. Rendering of Lilypond Markup

The storage of semantically meaningful musical data in MusicXML and presentation data in Lilypond markup mirrors the trend in the HTML or DocBook worlds, where structural information is kept independent of the final rendering. As a side benefit, these scripts also are useful in a pinch as a MusicXML to Lilypond to dvi/ps/pdf converter.

During the performance, the audience can and should input data that will direct the future course of the music. I wanted to use a GUI interface for the audience. The piece uses bash scripting as a glue language, but all of the real work is done by Python, making the various Python GUI options obvious candidates. For several reasons, I chose a Web browser interface with CGI forms for data input. What are the advantages of a Web interface over a Python-based GUI?

Web browsers are ubiquitous, among the most commonly used user applications in the world. Rather than requiring an installation of an X client and Python with GUI libraries and then needing to work out OS-specific issues, I simply can require performance sites to have machines with Web browsers—a trivial request.

It is largely due to browsers' ubiquity that they also are familiar. Self-described novice computer users are often far more comfortable operating a browser than some new GUI interface they've never seen before. In fact, even though page layout rendering of HTML often differs a great deal from one instance to another, it feels familiar to the user. This comfort level is especially critical for a performance situation like mine, where the audience needs to get over a natural reluctance to participate actively in a process traditionally reserved for the performers.

In addition, HTML is easy to code. Rob Pike's Rule 4 states that “fancy algorithms are buggier than simple ones”, and straightforward markup usually is even simpler. More information about the benefits of thin-client designs like this can be found in Hugh Williams and David Lane's *Web Database Applications*, published by O'Reilly and Associates.

Using a browser interface has clear drawbacks, but luckily, none are particularly relevant to my project. HTTP is slow, but I need to update only once per page of music, no faster than about once per minute. HTML/CSS style is relatively inflexible, but my design needs are simple. Video is problematic for pure HTML presentation without plugins that often are proprietary, but I need only still images.

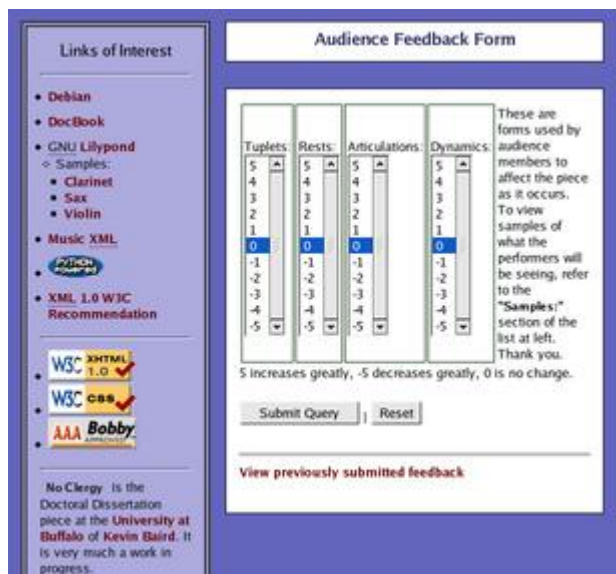


Figure 3. Sample User Interface

Given that we have a browser interface and hopefully a participating audience, we need to process their data. I use a Python CGI script to capture variables. It writes values within a `<pre>` tag, values that the later processing script reads in order to do the actual modification of musical data. This generic, extensible setup means that the capturing script needs little to no alteration as I improve the piece's other scripts by adding variables or changing how they are interpreted at other stages in the piece.

Listing 6. Script Fragment to Capture User Data

```
import cgi, re
form = cgi.FieldStorage()
formS = '<pre>\n'
for field in form:
    formS += field + ' = '
    formS += form[field].value + '\n'
formS += '</pre>\n'
```

The script then inserts the formS string at the appropriate point within the feedback file. The variables are integers representing percentage chances that a given note will have a given characteristic. I chose to list the values within `<pre>` tags to allow easy observation of changes to variables during a performance using a Web browser. The scripts always write the most recent data immediately after a `<!-- begin -->` comment.

In addition to the audience data variables, the processing script also incorporates a second-order Markov chain (see the "Markov Chains" sidebar). This allows all pages of music after the first to share characteristics of the first page while still being shaped by the audience feedback.

Markov Chains

Markov Chains, named after Russian Mathematician A.A. Markov, are ordered collections of data based on samples. For each set element x , we find all transitions from it to any element, including a repetition of itself. From those data, we find the probability that x leads to each element that follows it. We then can use those probabilities to reconstruct a randomly generated set that resembles the source set without necessarily being identical to it.

The user then continues to run the `noclergy.py` script for as long as desired. Another wrapper shell script could keep the piece running until a specific condition is met, such one of the `pc` variables reaching a given minimum or maximum. It also could have a set number of iterations. It even could run as a long-term installation. No performer relishes the thought of sitting for eight or more hours and playing whatever appears on a screen in front of him or her. Therefore, for very long runs, I should alter the program such that it does not require performers. One option would be to output to a synthesizer such as `Csound`. Another option would be to display the notation for visual demonstration only.

Future plans for this project include a port to the Ruby language. This is mainly for self education, but there are other reasons. Ruby code tends to be more compact—the advantages of features like the `each` method add up with enough code. In informal benchmarking, I've also found Ruby faster than Python for some tasks. I'd like to test this more scientifically, with and without Python bytecode optimization. Finally, by porting, I'll learn both Python and Ruby better—always a good thing.

As mentioned previously, the scripts store previous runs of the piece in the MusicXML format. There is no reason that other music in that same format couldn't be run through the script. By using other source music, the `mutate.py` script can create Markovian blends of Charlie Parker and Bartok, or whatever suits the user's tastes.

Finally, here's some technical trivia that may interest readers. I achieved much faster XML access when I did “dumb” file/string readline operations with my own convention for placement of `\n` characters. Conversion to standard DOM-aware XML libraries slowed the program down noticeably but made it both more robust and more usable by others. For a while, I even kept both versions of XML reading available. Eventually, I decided to use only the DOM-based reading, even though it has become the slowest portion of the script by far.

In the end, I felt that arguments for using an HTTP-based audience interface had merit for the XML access issue. Using the standard DOM library made the

script available to a wider user base. The XML read runs fast enough for usability today and will be even faster in the future with faster hardware. Therefore, I decided to comply with existing standards in the interest of robustness.

I plan to have a performance at the University at Buffalo in late 2004. Obviously, the program's collision with a real-world test should provide a great deal of useful information about how to improve it. The specifics of the performance are still in question, but I may try to use wireless browsers for the audience, which I hope will provide a less intimidating interface for them and will improve the overall experience for both performers and audience members.

Resources for this article: www.linuxjournal.com/article/7815.

Kevin Baird is pursuing a PhD in Music Composition at the University at Buffalo in Buffalo, New York. He is on-line at kevinbaird.net.

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

[Advanced search](#)

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

[Advanced search](#)

Beating Spam and Viruses with amavisd-new and Maia Mailguard

Robert LeBlanc

Issue #128, December 2004

Think you can't afford best-of-breed spam and virus protection for your business? Here are two good reasons to think again.

With spam and e-mail worms on the rise, it's boom-time for the makers of antispam and antivirus solutions. New anti-spam laws in Europe and the US have done little to solve the problem, and this situation has sent many people shopping for technological solutions: spam and virus filters.

Scanning and filtering content at every desktop is expensive and impractical, however. Ideally, the spam and virus problem should be tackled as close to the source as possible, to shield everyone downstream. This strategy lets an organization focus its resources on one place, typically the mail gateway.

Server-based solutions rarely come cheap, however. Most of these products are licensed on a per-mailbox basis, whether as add-on software for mail servers or as standalone content-filtering appliances. These solutions can cost thousands of dollars and often require annual subscription fees for access to updated virus signatures and spam patterns.

In this article, we take a look at an open-source content-filtering solution, amavisd-new, and a powerful extension of this project called Maia Mailguard.

amavisd-new

Conceptually, amavisd-new is a mail filter—it receives mail from your mail gateway, scans the mail for viruses and spam, quarantines, rejects or discards offending items, and relays the rest to another mail server downstream for delivery. In practice, amavisd-new often is sandwiched between two mail servers running on the same host, particularly at smaller sites where hosting

the mail server and content filters on a single machine is practical. Larger sites may choose to install amavisd-new, SpamAssassin and virus scanners together on a separate content-filtering machine. Massive sites may want a load-balanced array of such machines.

amavisd-new was written in Perl, with security and reliability in mind, and works well on virtually all UNIX platforms. It is an RFC-compliant mail handler, designed never to lose any mail. To that end, amavisd-new does not accept responsibility for a mail item until the downstream mail server has done so. This means any errors that occur while filtering the mail do not cause the mail to be lost; it remains in the upstream mail server's queue. amavisd-new offers four types of filtering: virus/malware scanning, spam filtering, banning dangerous attachment types and invalid mail headers.

Virus Scanning

amavisd-new is not a virus scanner; rather it's a framework that calls one or more virus scanners. More than 30 popular virus scanners currently are supported, including proprietary products from such vendors as Sophos, Symantec and Network Associates, as well as the open-source Clam Antivirus.

Both command-line and dæmonized virus scanners are supported, though dæmonized scanners are much more efficient than their command-line cousins. If your mail server processes a lot of mail, you don't want to have to load a command-line scanner into memory for each mail item and unload it afterward. A virus scanner that runs as a dæmon gets loaded once and then stays in memory, making the process much faster.

If you have multiple virus scanners installed, you can arrange them in primary and secondary groups. The secondary group is consulted if none of the primary scanners is operational.

Spam Filtering

Spam filtering is handled by amavisd-new by integrating it with SpamAssassin. amavisd-new calls SpamAssassin once per mail item, no matter how many recipients there are, so mailing-list postings don't consume any more resources than does mail addressed to a single recipient.

SpamAssassin provides a broad-spectrum approach to spam filtering, including feature recognition, DNSBL and SPF lookups, collaborative reporting networks and Bayesian learning mechanisms. All of these tests contribute a numeric score to a total for each mail item, and each user can specify a threshold score for deciding whether an item is spam or ham. This is an effective combination, as the strengths of one method make up for the weaknesses of another.

Feature recognizers check the headers or the body of the e-mail looking for patterns that human beings have identified as markers of spam or ham (non-spam mail). The fact that the Date: header contains a time 12 hours in the future or that the mail contains an image but no text might qualify as spam symptoms, whereas a message containing more than a thousand words is more likely to be ham.

SpamAssassin also can check the IP address of the connecting mail server or client against a number of DNS-based block lists (DNSBLs) to determine whether that address is a known spam source. Unlike the traditional use of DNSBLs, however, SpamAssassin does not consider a listing to be damning by itself; it simply adds a value to the mail's total score. This is a much more flexible approach, one that lets you adjust the scores assigned to each DNSBL according to how much you trust that list and the policies of its maintainers. The upcoming SpamAssassin 3.0 also adds support for Sender Policy Framework (SPF) lookups, which try to verify that the connecting host has the authority to send mail for its domain.

Collaborative reporting networks, such as Vipul's Razor, Pyzor and the Distributed Checksum Clearinghouse (DCC) offer another kind of resource for SpamAssassin to consult. The idea is that because spam is broadcast to millions of recipients, by the time you receive your copy, a lot of other people have received more or less identical copies. If a lot of those people already have reported that particular mail as spam, your own spam filter should be able to use that fact in its own decision-making process.

Last, but certainly not least, SpamAssassin offers a Bayesian learning mechanism, which essentially is an automated feature recognizer. Although the manually designed feature recognizers listed above rely on human beings to point out features that indicate spam or ham, the Bayesian approach tries to pick out these features automatically, based on an analysis of the spam and ham you've received already.

Banning Dangerous Attachment Types

As a fail-safe measure, it often is a good idea to block mail containing executable attachments, even though your virus scanners may claim they're clean. Virus scanners aren't perfect, after all, and brand-new malware might reach your network before your antivirus vendor makes a new signature available to detect it. amavisd-new lets you define a list of file extensions, content classes and MIME-types that should be quarantined, rejected or discarded.

Dealing with Invalid Mail Headers

According to RFC 2822, mail headers are not supposed to contain any characters above 127 nor any NUL or bare carriage-return characters. Characters outside this range are supposed to be specially encoded, so that mail software around the world can parse them without confusion. When mail with invalid headers arrives, it could be the product of a poorly written mail client, but often this is a symptom of a specially designed program used by spammers to do their mass mailings. The authors of this so-called ratware often are English speakers, and they don't typically think about the fact that their software might be used by spammers who speak other languages. When those spammers try to use these tools to send their mail, the ratware does not encode the special characters, producing invalid mail headers. With amavisd-new, you can decide how to handle mail with invalid headers: quarantine it, reject it, discard it or let it through.

Setting Content-Filtering Policies

amavisd-new lets administrators define system-wide content-filtering policies, but these settings can be overridden at the domain and user levels. Some users may want to have their mail scanned for all four suspicious content types—viruses, spam, banned files and bad headers—while others might prefer to disable one or more of those checks. One user might want mail arriving with a spam score of 5.0 or higher to be quarantined, while another user might prefer to have the Subject: header prefixed with a special tag, such as *****SPAM*****, if the score is 4.0 or higher but have it blocked only if the score is at least 8.0. This fine-grained control over the filtering process lets administrators accommodate a wide range of users with different needs.

Similarly, amavisd-new provides whitelists and blacklists at all three of these levels. This allows administrators to define system-wide lists; at the other end, users can maintain their own individual lists.

Quarantining and Notification Options

When amavisd-new blocks an e-mail, it can be configured to do a number of things to that mail. The mail can be stored in a quarantine directory or mailbox, including special per-user mailboxes, such as joe+spam. You also can configure amavisd-new to reject the mail, refusing to accept it from the upstream mail server or discard it quietly.

If your organization's policies require that you notify the senders of blocked mail, amavisd-new can be configured to do so. This is a controversial subject, however. A lot of people find virus alerts and spam complaint e-mails to be more of a nuisance than a help nowadays, particularly because the sender

addresses of these items often are forged. If you must send virus notifications, amavisd-new provides a mechanism for listing the viruses known to fake the sender's address, so notices are not sent out when those viruses are detected. This list must be maintained by hand and must be matched to the names your particular virus scanners generate. If you find it easier to list the viruses that don't fake sender addresses, you can use an inverse list instead.

Maia Mailguard

The Maia Mailguard Project began its life as a simple Web front end for amavisd-new, designed to let users adjust their content-filter settings and manage their quarantines from a convenient interface. The project proved quite popular with ISPs, Web-mail providers and companies offering off-site content filtering, however, and the needs of these larger-scale clients soon developed Maia Mailguard into something much more sophisticated.

Maia Mailguard is a complete spam and virus management system, consisting of PHP, SQL and Perl scripts, a MySQL or PostgreSQL database and, of course, amavisd-new, SpamAssassin and supported virus scanners. Arrays of content filters can be managed from a single Maia interface, all sharing the same SQL database. Designed to make content filtering, quarantine management and spam reporting easier, Maia Mailguard is in many ways a new kind of tool for mail users.

Web Interface

Maia's Web-based interface lets users authenticate against a variety of sources, including a POP3 or IMAP server, an LDAP server, an external SQL database or Maia's own internal database. Users can be added manually by an administrator or automatically when mail arrives for a local address that Maia hasn't seen before.

Users can have multiple e-mail addresses linked to their accounts, but each e-mail address has its own content-filtering settings (Figure 1). Users can add and remove addresses from their whitelists and blacklists with the Web interface (Figure 2), while administrators manage domain-level and system-wide settings from another set of Web pages (Figure 3). Statistics are maintained for all four of amavisd-new's mail types, as well as blacklisted and whitelisted items, oversized items, false positives and false negatives (Figure 4). Other tables keep track of viruses by type and by how often specific SpamAssassin rules are triggered. Graphical charts can be generated on the fly from this data or generated as static pages at scheduled intervals.

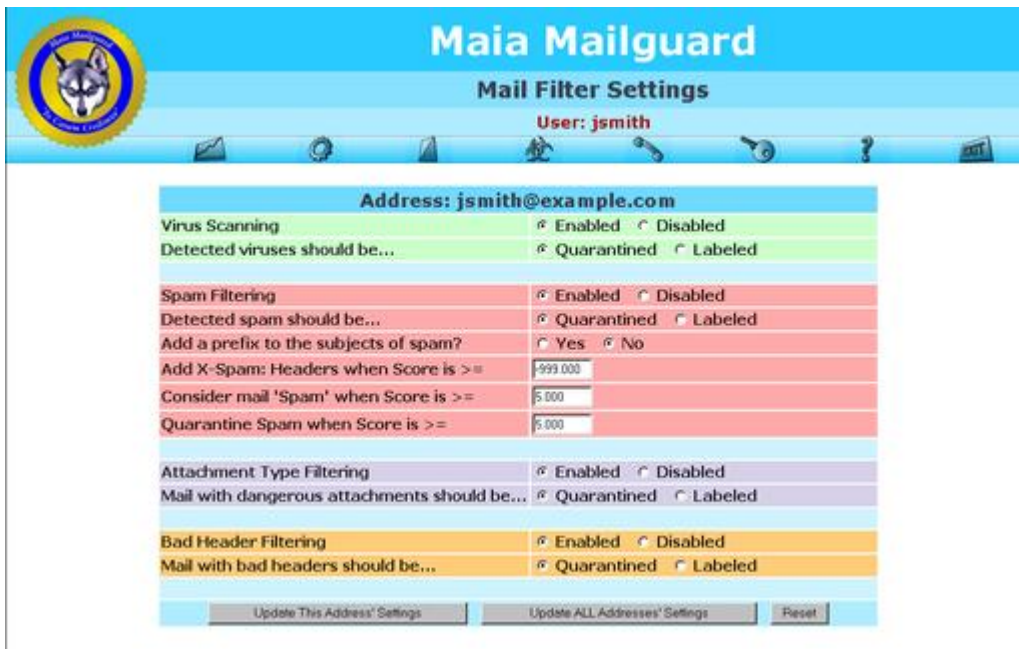


Figure 1. Every e-mail address has its own content-filter settings.

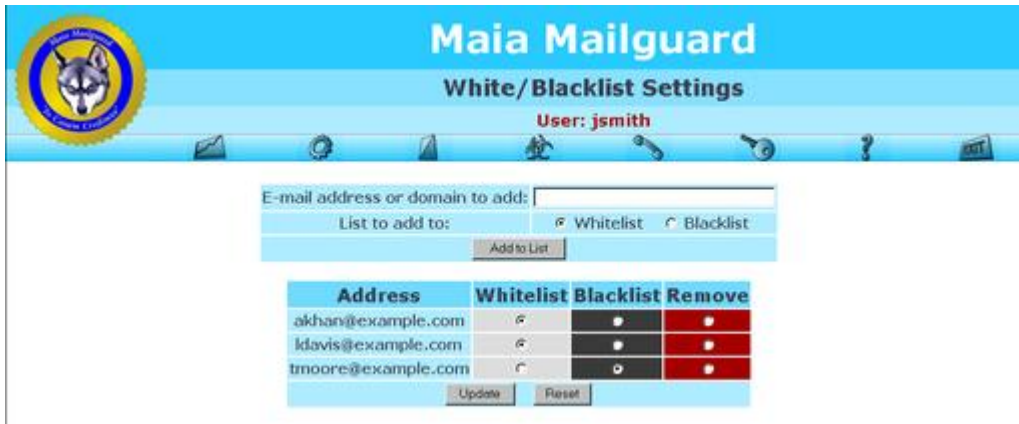
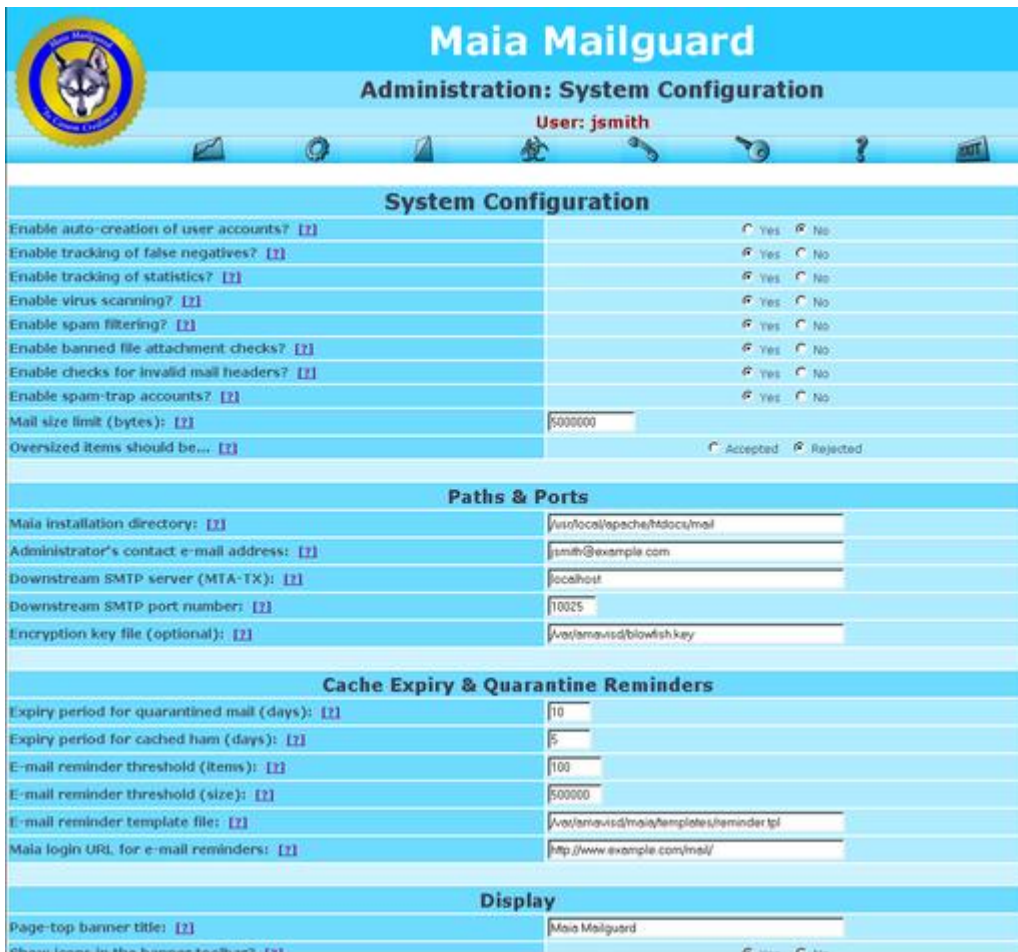


Figure 2. Users maintain their own whitelists and blacklists.



Maia Mailguard
Administration: System Configuration
User: jsmith

System Configuration

Enable auto-creation of user accounts? Yes No
 Enable tracking of false negatives? Yes No
 Enable tracking of statistics? Yes No
 Enable virus scanning? Yes No
 Enable spam filtering? Yes No
 Enable banned file attachment checks? Yes No
 Enable checks for invalid mail headers? Yes No
 Enable spam-trap accounts? Yes No
 Mail size limit (bytes):
 Oversized items should be... Accepted Rejected

Paths & Ports

Maia installation directory:
 Administrator's contact e-mail address:
 Downstream SMTP server (MTA-TX):
 Downstream SMTP port number:
 Encryption key file (optional):

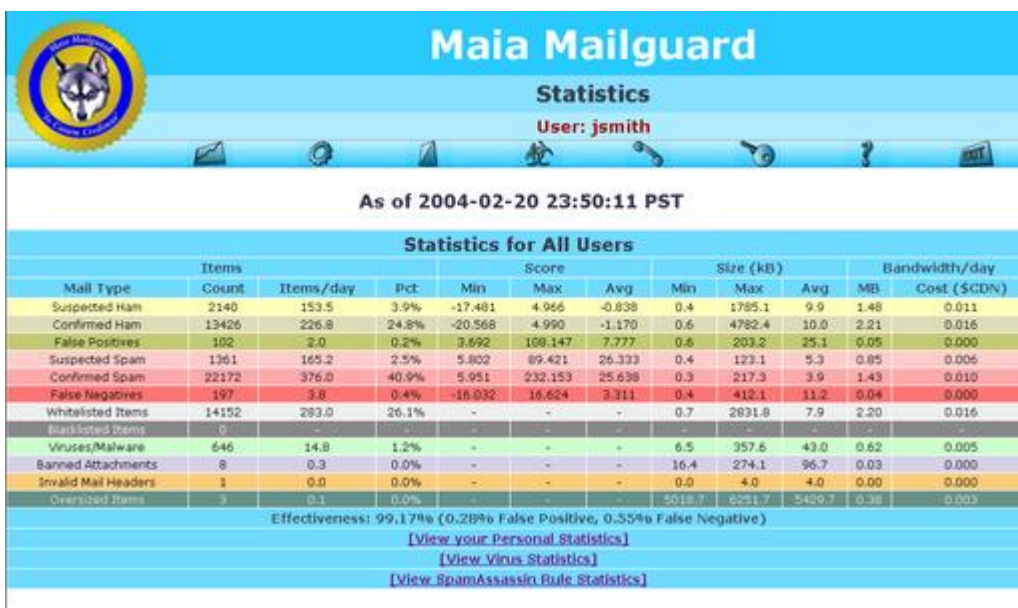
Cache Expiry & Quarantine Reminders

Expiry period for quarantined mail (days):
 Expiry period for cached ham (days):
 E-mail reminder threshold (items):
 E-mail reminder threshold (size):
 E-mail reminder template file:
 Maia login URL for e-mail reminders:

Display

Page-top banner title:

Figure 3. The administrator can configure most global settings from the Web interface.



Maia Mailguard
Statistics
User: jsmith

As of 2004-02-20 23:50:11 PST

Statistics for All Users

Mail Type	Items			Score			Size (kB)			Bandwidth/day	
	Count	Items/day	Pct	Min	Max	Avg	Min	Max	Avg	MB	Cost (\$CDN)
Suspected Ham	2140	153.5	3.9%	-17.491	4.966	-0.838	0.4	1785.1	9.9	1.48	0.011
Confirmed Ham	13426	226.8	24.8%	-20.568	4.990	-1.170	0.6	4782.4	10.0	2.21	0.016
False Positives	102	2.0	0.2%	3.692	108.147	7.777	0.6	203.2	25.1	0.05	0.000
Suspected Spam	1361	165.2	2.5%	5.802	89.421	26.333	0.4	123.1	5.3	0.05	0.006
Confirmed Spam	22172	376.0	40.9%	5.951	232.153	25.638	0.3	217.3	3.9	1.43	0.010
False Negatives	197	3.8	0.4%	-16.032	16.624	3.311	0.4	412.1	11.2	0.04	0.000
Whitelisted Items	14152	283.0	26.1%	-	-	-	0.7	2831.8	7.9	2.20	0.016
Blacklisted Items	0	-	-	-	-	-	-	-	-	-	-
Viruses/Malware	646	14.8	1.2%	-	-	-	6.5	357.6	43.0	0.62	0.005
Banned Attachments	8	0.3	0.0%	-	-	-	16.4	274.1	96.7	0.03	0.000
Invalid Mail Headers	1	0.0	0.0%	-	-	-	0.0	4.0	4.0	0.00	0.000
Oversized Items	3	0.1	0.0%	-	-	-	5018.7	6251.7	5429.7	0.38	0.003

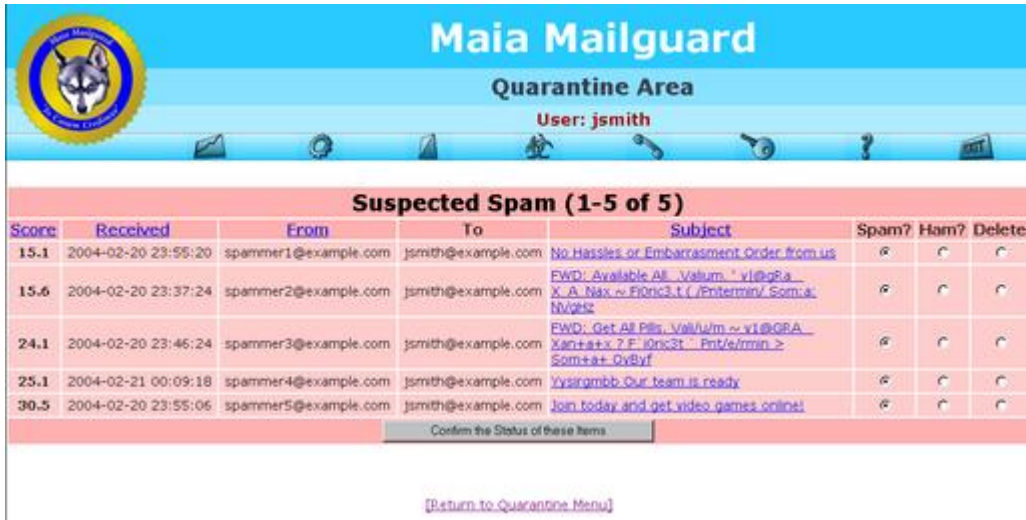
Effectiveness: 99.17% (0.28% False Positive, 0.55% False Negative)
[\[View your Personal Statistics\]](#)
[\[View Virus Statistics\]](#)
[\[View SpamAssassin Rule Statistics\]](#)

Figure 4. The stats table summarizes what your filters have seen.

Thanks to the fact that Maia puts quarantine management and content-filtering controls in the hands of users themselves, there isn't a lot of work left for administrators to do on a day-to-day basis. With Maia's Perl scripts running at scheduled intervals to report user-confirmed spam and to expire old quarantine items, the system all but manages itself.

Quarantine Management

When mail gets quarantined on behalf of a user, it's important that the user has a convenient way to access that mail. Maia provides a list of the items in a user's quarantine, sorted by spam score so that the items most likely to be there by mistake—the false positives—are kept closer to the top of the list and are easier to spot (Figure 5).



Score	Received	From	To	Subject	Spam?	Ham?	Delete
15.1	2004-02-20 23:55:20	spammer1@example.com	jsmith@example.com	No Hassles or Embarrasment Order from us	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
15.6	2004-02-20 23:37:24	spammer2@example.com	jsmith@example.com	FWD: Available All Valum ~ vl@p8A... X_A_Nax ~ F!0nc3.t (/Entermin/ Som:a... Nv9htz	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
24.1	2004-02-20 23:46:24	spammer3@example.com	jsmith@example.com	FWD: Get All Pills Valium ~ vl@Q8A... Xan+ + x 7 F' !0nc3t ~ Pnt/a/rmin >... Som+ + QvByf	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
25.1	2004-02-21 00:09:18	spammer4@example.com	jsmith@example.com	Vysrgmbb Our team is ready	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
30.5	2004-02-20 23:55:06	spammer5@example.com	jsmith@example.com	Join today and get video games online!	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Confirm the Status of these Items

[\(Return to Quarantine Menu\)](#)

Figure 5. The user's quarantine is sorted by spam score.

If you're not sure from the subject line whether the mail is legitimate, you can click on the subject to open the e-mail in Maia's mail viewer (Figure 6). The mail viewer is safe to use on all types of mail, as it doesn't decode most attachments but does block remote images and strip away HTML tags that could redirect you to another site. You can view the mail in its decoded form or in its raw form, complete with all of the original mail headers.

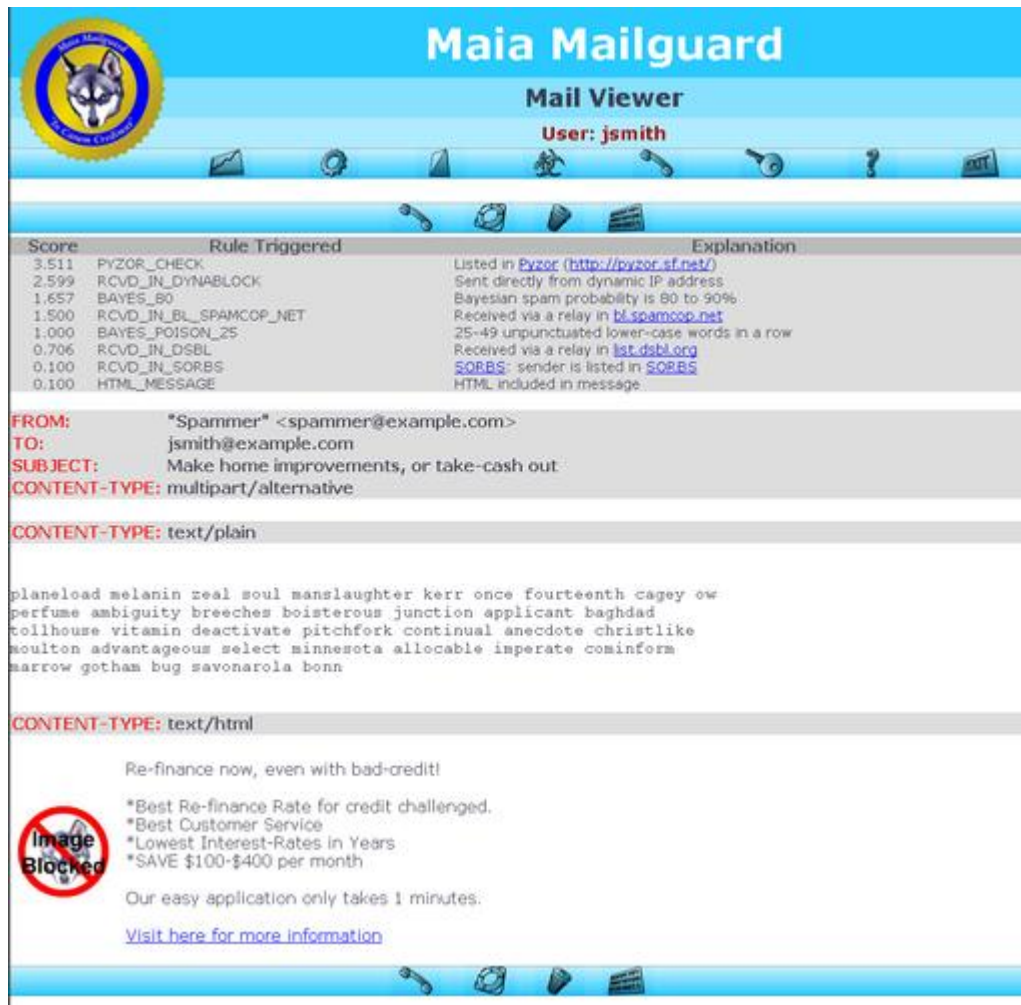


Figure 6. The mail viewer provides a safe way to inspect suspicious mail.

If you decide that the mail is legitimate after all, you can click a button to rescue the item from your quarantine and have it delivered to you. At the same time, Maia tells SpamAssassin about the mistake; the Bayesian learning system is less likely to make the same mistake in the future. You also can configure Maia to add the sender's address to your whitelist automatically when you rescue an item in this manner.

In addition to the quarantine, Maia offers a ham cache, which essentially is a list of the legitimate mail that you've received recently (Figure 7). The purpose of the ham cache is to let you report spam that somehow got past the filters—the false negatives. By marking these items properly as spam, you help to train SpamAssassin's Bayesian learning system.



Figure 7. The ham cache allows a user to report false negatives.

The quarantine and ham cache also provide a means for you to confirm the status of the mail you've received. This not only helps train the Bayesian learning system, it also makes it possible to report spam properly, because it's been confirmed by a human being.

Spam Reporting

Most spam filters are concerned only with defending against the onslaught of spam and do little or nothing to prevent it in the first place. Because Maia allows users to confirm the status of their mail as spam and does nothing to modify the original mail headers, this spam can be reported in a number of different ways. Upcoming versions of Maia will support detailed header analysis and semi-automated reporting to ISPs. These reports help others block spam more effectively and even can result in some form of punishment for the spammer.

Behind the scenes, Maia's automated scripts process the quarantine at regular intervals, reporting confirmed spam to the same collaborative networks that SpamAssassin consults—Vipul's Razor, Pyzor and the DCC. By sharing this information with these networks, you give something back, rather than only benefiting from the reports of others.

An Effective, Comprehensive Solution

In the end, what matters most is how effective the combination of amavisd-new and Maia Mailguard is at keeping spam out of your inbox, while keeping ham out of your quarantine. From my own site's statistics, that figure is a refreshing 99.22%, with 0.26% false positives and 0.52% false negatives. Best of all, those false positives can be recovered easily from the quarantine and the false negatives can be reported from the ham cache.

For viruses and other forms of malware, the effectiveness figure is even more impressive: 100%. In the six months since I installed this content-filtering

solution, the virus scanners on my desktop machines haven't caught anything that slipped past the filters. This is largely due to the way amavisd-new allows multiple virus scanners from different vendors to be used together—what one scanner misses, another typically catches.

Performance-wise, any content-filtering solution is going to slow down mail processing to some extent. It often becomes a trade-off between filter effectiveness and speed, as you may choose to disable certain filters and tests to improve mail throughput. My 99.22% effectiveness statistic comes from having every available test and filter enabled, for example, but it also costs 1–3 seconds to process each mail item on a moderately loaded dual-PIII 733MHz with 1GB of RAM. A busier site might not be able to tolerate that kind of delay. They would have to choose between disabling the more time-consuming tests, upgrading the processor and RAM in the content filter and switching to a load-balanced array of content filters. Nevertheless, Maia Mailguard and amavisd-new are being used together at sites hosting more than 50,000 users, processing more than 350,000 e-mails a day, so the solution scales if you've got the hardware to handle it.

As many people already have discovered, some of the best weapons in the war against spam and viruses happen to be open-source tools. With tools like amavisd-new, Maia Mailguard, SpamAssassin and Clam Antivirus, you can provide your network with world-class protection without spending tens of thousands of dollars.

Resources for this article: www.linuxjournal.com/article/7820.

Robert LeBlanc is the president of Renaissoft (www.renaisssoft.com), author of Maia Mailguard and resident spam-fighting guru for the AnswerSquad (www.answersquad.com). When he's not reinventing the wheel or building better mousetraps, he can be found in the company of his four Alaskan Klee Kai, Zorro, Sikari, Piyomi and, of course, Maia.

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

[Advanced search](#)

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

Revision Control with Arch: Maintenance and Advanced Use

Nick Moffitt

Issue #128, December 2004

Put a powerful revision control system to work for you with only Web and SSH software on the server side. Here's what you need to administer a software project with Arch.

Arch is part of a recent generation of revision control systems that provide an important architectural advantage over the old Concurrent Version System (CVS) and its work-alikes. As a decentralized revision control system, Arch allows remote users to join large development efforts without needing to acquire special access privileges. Arch also provides powerful inter-archive operations that encourage participation from third-party contributors.

The previous article in this series [[LJ](#), November 2004] demonstrated basic Arch operations, such as checking out code and creating branches from remote archives. This installment shows how to revert changes in an archive, how to publish your private archives to public mirrors and how to move a copy of your changes from archive to archive when you forget to make a new branch.

The Arch program is called `tla`. The program name `arch` is taken by the POSIX standard, which requires that `/bin/arch` report system information. A lot of information can be found by running `tla help`. If you need to figure out the arguments to a particular command, such as `commit`, it helps to run `tla commit -H`, to see what the `tla commit` command can do.

Backing Out Changes

One of the more immediate benefits of any revision control system is the ability to undo a change or set of changes. Everyone makes mistakes now and again, and it is important for your tools to provide the means to a graceful recovery.

The quickest way to return a checked-out tree to a state without your local changes is to run `tla undo`. This creates a directory called `„undo-1/` that contains all of the changes made. If you so desire, you simply can `tla redo` to re-apply those changes. For example:

```
$ tla register-archive http://www.lnx-bbc.org/arch
$ tla get \
  lnx-bbc-devel@zork.net--gar/lnx-bbc--stable bbc
$ cd bbc/
$ echo "BIG MISTAKE" > robots.txt
$ echo "#smaller change" >> Makefile
$ tla undo
$ tla redo
```

The `tla undo` command is most useful during hold-that-thought moments, when a line of work needs to be set aside briefly for a quick change of some sort. Arch uses the `undo` and `redo` commands internally when performing operations such as `update` or `star-merge`.

Reverting One File

If a mistake is localized to a single file, the entire changeset doesn't need to be backed out. Arch lets you revert the changes made to a single file by generating a unified diff representing that file's changes since the last commit. This diff then can be fed into the `patch` program in reverse mode, which causes the changes to be unpatched out of the file.

```
$ tla file-diffs robots.txt | patch -R
```

If the file had been deleted accidentally, it would be necessary to do `touch robots.txt` before executing this command. Without a file (even an empty one), Arch has no basis from which to generate the file-diffs. When working with complete changesets, however, Arch is far more intelligent.

Reverting Entire Changeset

One of the big advantages Arch has over its predecessor, CVS, is that it permits the creation and manipulation of changesets. A changeset is a complete collection of all the edits, renames, added and deleted files and log entries recorded during a single `tla commit` invocation.

Sometimes a changeset is committed that shouldn't be, or a temporary approach to something needs to be backed out before a more permanent one can be implemented. In these cases, revert the changeset by replaying it in reverse:

```
$ tla replay --reverse \
```

```
jrhzork.net--projects/foo--bar--1.0--patch-4
$ tla sync-tree \
jrhzork.net--projects/foo--bar--1.0--patch-4
```

The first command reverts the fourth changeset in the 1.0 version of the bar branch of the foo tree, even if it is not the most recent revision. This has the added effect of backing out the log entry for that changeset as well, so you can use the `tla sync-tree` command to put the commit log back the way it ought to be.

The patch-4 changeset still is stored in the jrhzork.net—projects archive, and the tree still can be checked out in that state. Only the current working copy of the code has been affected by the above commands. When the above user runs `tla commit`, a new changeset will be added that includes the inverse of patch-4.

Cherry-Picking Changes from Another Branch

The `tla replay` command can be used for more powerful operations than a simple `undo`. One of the more compelling features of Arch is the ability to cherry-pick particular changesets from a remote archive without having to apply changes you don't need.

Consider the project, foo, maintained by Bob. Bob keeps a stable branch of the project (foo--stable) and an experimental branch (foo--experimental). All releases are generated from the stable branch—foo--stable--2.4.2 being the most recent. The experimental branch is where adventurous new features are made available in a somewhat official location.

Alice plans to work on some experimental code, so she tags off Bob's experimental branch to work in her own space:

```
$ tla my-id "Alice B. Hacker <abh@zork.net>"
$ tla make-archive -l abh@zork.net--work \
  sftp://abh@zork.net/home/abh/public_html/arch
$ tla archive-setup foo--hackery--0.0
$ tla register-archive http://entar.net/~bob/fooarch
$ tla tag \
  bob@entar.net--code/foo--experimental--0.0 \
  abh@zork.net--work/foo--hackery--1.0
```

In the process of working on her experimental features, Alice discovers a bug that Bob must have overlooked. The fix is simple, so she puts her current work aside with `tla undo` and checks in the fix:

```
$ tla undo
$ vi buggy_file.c another_buggy_file.c
```

```
$ tla commit
M  buggy_file.c
M  another_buggy_file.c
*  committed
   abh@zork.net--work/foo--hackery--1.0--patch-9
$ tla redo
```

Alice soon finishes her changes and tells Bob where her archive lives. Bob decides that her code is acceptable for the experimental branch and star-merges it in:

```
$ tla get bob@entar.net--code/foo--experimental--0.0
$ cd foo--experimental--0.0/
$ tla register-archive http://zork.net/~abh/arch/
$ tla star-merge \
  abh@zork.net--work/foo--hackery--1.0
```

While reading Alice's changelog, Bob realizes the bug she fixed exists in the stable branch as well. Because he doesn't want to grab all of the experimental code from her hackery branch, Bob cherry-picks only the changeset that contains the bug fix:

```
$ tla get bob@entar.net--code/foo--stable--2.4.2
$ cd foo--stable--2.4.2/
$ tla replay \
  abh@zork.net--work/foo--hackery--1.0--patch-9
```

Publishing Your Changesets

Alice and Bob were able to work together despite the fact that neither developer shared access to a single system. Neither developer had set up any sort of dedicated server; they were able to use standard stock protocols such as HTTP, SSH and SFTP. Alice's archive had the advantage of being accessible from a Web directory on the Internet, just as Bob's official archive was.

Arch provided the tools for Alice and Bob to manipulate their two separate archives, and the differences between them, using nothing more exotic than Apache and OpenSSH.

Signatures

Sending so much code over the Internet always has made free software developers at least a little nervous, even if only in the back of their minds. The current system of peer review seems to have solved the problem of malicious code submissions quickly and effectively, but it would help to be able to identify each changeset's author beyond a reasonable doubt.

Arch allows developers to sign their changesets cryptographically, allowing verification of submitter identity through a web of trust. Although this does not conclusively prove the intentions of the developer in question, it raises the bar for forged submissions.

To use cryptographic signatures in Arch, you first must generate a GnuPG key.

```
$ gpg --gen-key
```

Unfortunately, signed archives are somewhat different functionally from the unsigned variety. This makes it necessary to keep a separate archive for signed commits. Running `tla make-archive` with the `-s` switch creates an archive capable of storing GnuPG signatures:

```
$ tla make-archive -ls jrh@zork.net--signed \  
~/SIGNED-ARCHIVE  
$ tla my-default-archive jrh@zork.net--signed
```

Finally, a few configuration files must be created in order for Arch to sign changesets and verify signatures. First, an awk script included in the tla distribution, called `gpg-check.awk`, must be installed somewhere on the system where Arch is run. The Debian tla packages install it to `/usr/bin/tla-gpg-check` by default. In order for Arch to verify signatures, the file `~/.arch-params/signing/=default.check` should contain a single line that reads:

```
$ mkdir ~/.arch-params/signing/  
$ echo \  
'tla-gpg-check gpg_command="gpg --verify-files -" \  
> ~/.arch-params/signing/=default.check
```

If you want keys to be downloaded automatically from a public keyserver as needed, you can add parameters such as `--keyserver pgp.mit.edu --keyserver-options auto-key-retrieve` to the `gpg_command`. This causes Arch to download keys from pgp.mit.edu as needed and verify the signatures in an archive against these keys during the `get` or `update` operations.

For Arch to sign changesets automatically that you commit to an archive created with the `-s` option, the `~/.arch-params/signing/=default` file must be one single line like the following, substituting the address you used when you created your key:

```
$ echo \  
'gpg --default-key "<jrh@zork.net>" --clearsign' \  
> ~/.arch-params/signing/=default
```


Mirrors

In the above cherry-picking example, Alice B. Hacker used a Web-accessible directory for her personal archive. This is convenient, but it poses a problem for disconnected use. What if Alice wanted to work from her laptop during a long airplane flight or train ride? She either would have to generate changeset tarballs with `tla changes` or star-merge her various branches manually one by one from her laptop to her Web-space archive when she reached a network connection. Fortunately, Arch permits the creation of archives that are simply mirrors of other archives:

```
$ tla make-archive -ls --mirror-from \  
  jrh@zork.net--signed \  
  sftp://jrh@zork.net/public_html/arch/
```

In this instance of `make-archive`, J. Random Hacker is creating an archive in his `public_html` directory on an Internet server. Once the mirror archive is created, it shows up in a `tla archives` listing as `jrjrh@zork.net--signed-MIRROR`. Now data can be pushed to it with a single command:

```
$ tla archive-mirror jrjrh@zork.net--signed
```

In addition to push mirrors that copy local archive data to remote systems, Arch allows pull mirrors that create local copies of remote archives:

```
$ tla make-archive -ls --mirror \  
  lnx-bbc-devel@zork.net--gar \  
  /var/tmp/gar-cache  
$ tla archive-mirror lnx-bbc-devel@zork.net--gar
```

This can be handy during disconnected operation, when a local branch may not be sufficient. Pull mirrors allow read-only access to a remote archive's data while off the Net.

Signed Mirrors

One drawback to the `jrjrh@zork.net--signed-MIRROR` archive is that it is a separate signed archive in its own right. This means J. Random Hacker must sign each changeset as it is copied from the original archive to the mirror.

In some cases, this is the desired effect. A release manager personally vouches for each changeset that enters the public mirror, for example. In most cases, however, it is important simply to copy the existing signatures along with the

changeset. This is achieved by creating a special file on the system where `tla archive-mirror` is run:

```
$ echo jrh@zork.net--signed > \  
~/arch-params/signing/jrh@zork.net--signed-MIRROR
```

Working Off-line (Even If You're Absent-Minded)

Mirrors are extremely useful, but they are, by nature, read-only. The only way changes can be committed to a mirror is through the original archive by way of `tla archive-mirror`.

Consider Alice's laptop mirror situation. While sitting in the observation car of Amtrak's Coast Starlight, she pulls out her laptop and does `tla get` to grab some code out of a local mirror of `abh@zork.net--work`. Somewhere in the Willamette Valley, she finds inspiration and completes a remarkably useful hack.

Any attempt to commit her changes would receive the message `attempt to write directly to mirror`, which means the commit failed. The simple solution is to wait until she reaches an Internet access point and use the `undo` and `redo` commands:

```
$ tla undo ,changes-to-mirror  
$ cd ~/real-project/  
$ tla redo ~/mirror-checkout/,changes-to-mirror/  
$ tla commit
```

This works fine if your changes are not enough to require more than one changeset. For longer detached sessions, you'll want to make a new local branch.

After her trip down the Pacific Coast, Alice takes the Zephyr to Chicago. It is a longer trip, and she found herself working in a local mirror of `bob@entar.net--code` on the `foo--stable--2.4.2` code. After a few hours of work, she decides to move her changes to a new branch.

First, she makes a new archive and branch on her laptop:

```
$ tla make-archive -l abh@zork.net--laptop ~/arch  
$ tla my-default-archive abh@zork.net--laptop  
$ tla archive-setup foo--laptop-hacks--1.0
```

Next, she tags off the mirror branch to her new archive. She runs the `tla Logs` command in shell backticks so she doesn't have to remember which patch level and version she was working in at the moment:

```
$ tla tag `tla logs -r -f | head -n 1` \  
foo--laptop-hacks--1.0
```

Finally, Alice coerces the checked-out copy into believing it is the first revision in her new laptop-hacks branch:

```
$ tla sync-tree foo--laptop-hacks--1.0--base-0  
$ tla set-tree-version foo--laptop-hacks--1.0
```

At this point, she has shifted her checked-out copy from the read-only mirror over to a read-write archive hosted on her laptop.

Branching without an Archive

Setting up mirrors before long disconnected sessions is a lot like packing for a trip: you always forget the one thing you really needed. It would be frustrating to plug your laptop in to the light socket of your mountain cabin only to find that your checked-out copy of some crucial code came from an HTTP archive.

Fortunately, you can use some of the same techniques to move a checked-out copy to a new branch even if you can't reach the old read-only archive.

Alice checked out a copy of a project called bar while sitting in an Internet café in Chicago. On her return trip to California, she decides to work on the code. After another hour of prodigious efforts, she decides yet again that it is time to make her own branch in which to work.

Because the original archive is inaccessible, tagging off a branch is impossible. Fortunately, much of the changelog and history information is present in the checked-out tree, so Alice temporarily backs out her changes with `tla undo` and then forces the checked-out copy into her new branch:

```
$ tla archive-setup bar--train-ride--1.0  
$ tla set-tree-version bar--train-ride--1.0  
$ tla add-log-version bar--train-ride--1.0  
$ tla import
```

Once this is done, Alice runs `tla redo` and then `tla commit`. Now the revision she grabbed in Chicago is bar--train-ride--1.0--base-0, and her changes are bar--train-ride--1.0--patch-1.

Although this method is not perfect, it still is possible to star-merge to and from the original branch without trouble. If Alice found her work on the bar project to be more involved, she most likely would merge with the upstream archive and make a proper branch when she found Internet access again.

Tune in Next Time

You now know how to publish your archives to the Internet and how to work remotely with Arch. You even have a few tricks up your sleeve for when you make mistakes,

The third and final article in this series will show you how to administer a centralized official archive while retaining all of the benefits of Arch's distributed workings. You'll learn some tricks for scripting around your archives to create reports on development activity, as well as the creation of a test build infrastructure.

Nick Moffitt is a Linux professional living in the San Francisco Bay Area. He is the build engineer for the LNX-BBC Bootable Business Card distribution of GNU/Linux and the author of the GAR build system. When not hacking, he studies the history of urban public transportation.

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

[Advanced search](#)

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

[Advanced search](#)

Automating Manufacturing Processes with Linux

Craig Swanson

Ryan Walsh

Issue #128, December 2004

How do you monitor multiple manufacturing lines in real time? We did it ourselves with Linux.

A manufacturing company makes money only when production is running. So, timely information from the production floor is crucial to the business. As our company has grown, so has the complexity. We have outgrown our manual and paper methods for monitoring manufacturing.

Midwest Tool & Die stamps electronic terminals and molds plastic parts for the automotive, electronics and consumer industries. Our manufacturing processes generate a lot of data. Our high-speed presses make up to 1,200 parts per minute, and each part must be correct. We inspect critical dimensions for every part that is produced. Part quality is charted and monitored, and the data is archived for traceability.

Why Automate?

We needed to manage all of this data to improve the manufacturing processes. Our main goals were to improve uptime and understand the causes for downtime. In addition, we hoped to track and control costs, reduce paperwork and avoid human input error.

To meet these goals, we came up with requirements for the new system. The first requirement was to gather data from a variety of machine controls, sensors, automated inspection equipment, programmable logic controllers (PLC) and human operators. The system had to be reliable and be capable of gathering data at our fastest production rate.

Next, the system had to correlate the data that was gathered. The system would need to interact with enterprise PostgreSQL databases. Production data and process status would be passed to PostgreSQL for display and reporting.

The new automation system also had to provide a user interface, so the machine operator and maintenance personnel could log their activities. Process downtime and the reason for the downtime would be logged and passed to the enterprise databases. This requirement would replace a paper log and manual data entry effort.

Finally, the system needed to be flexible and easily upgraded. The solution would be adaptable to new manufacturing lines and changing system inputs.

Why Linux?

We evaluated several solutions to meet the requirements. Industrial PLCs could gather data reliably. However, their approach to networking has been stuck in proprietary nonstandards for decades. Ethernet connectivity has become available, but the systems are expensive. The user interface typically is implemented on vendor-specific display hardware. Each vendor produces its own proprietary development platform. So, vendor lock-in was an issue at every point of the evaluation.

Next, we looked at a PC with a data acquisition (DAQ) board. In the past, we have used a laptop with a DAQ board, Microsoft Windows and Agilent VEE. This combination has worked well for quickly acquiring data with little programming. With that setup, data transfer to our database systems was available only through Windows OLE. We could develop applications, but the proprietary platform would tie us in to the vendor. National Instruments also offers a complete DAQ package for the PC, but at a premium price.

The solution that best met our requirements also used a PC and a DAQ board. The big difference was the use of RTLinux, a real-time OS stack based on Linux. We could limit vendor tie-in and communicate freely with PostgreSQL and TCP/IP networking. The real-time OS offered the reliability of a PLC, without sacrificing communications. Finally, the GUI could be written in the language of our choice. Using open-source tools, we could create flexible, upgradeable applications.



Figure 1. Two SmartPress manufacturing automation systems in use. Each system includes a PC with DAQ board, electrical isolation hardware, battery backup and a barcode printer.

SmartPress System

The computers we chose to perform the task of data acquisition and handling of the data were slow in comparison to computers available today. We were able to recycle old office computers to use on the shop floor. These 400MHz Celeron machines are fast enough to perform the tasks asked of them adequately, without hindering our hard real-time requirements for data acquisition. The system we worked with started off with an installation of the Red Hat 7.3 distribution with the 2.4.18 Linux kernel.

RTLinux and Linux kernels

The kernel separates the user-level tasks from the system hardware. The standard Linux kernel allots slices of time to each user-level task and can suspend the task when the time is up. This can lead to priority tasks being delayed by your system's noncritical tasks. There are commands to control the operations of the Linux scheduler; however, hacking the scheduler's parameters in the 2.4 kernel does not result in a hard real-time system. The 2.6 kernel has enhanced real-time performance but does not fill the needs of a hard real-time system either.

There are many great publications about RTLinux, many of which are written by Michael Barabanov and Victor Yodaiken, who first implemented RTLinux back in 1996 and have been improving it ever since. Finite State Machine Labs, Inc.,

(FSMLabs) is a privately held software company that maintains the software. Through the years they have produced advancements that are wrapped up into their professional versions of RTLinux. They still do provide RTLinux/Free, which must be used under the terms of the GNU GPL and the Open RTLinux Patent License. For our project, we used the free software, which does not have support from FSMLabs.

The RTLinux HOWTO, by Dinil Divakaran, provided the majority of the information we needed to complete the RTLinux installation and get it up and running on our system.

RTLinux System Flow

In a standard Linux system, if you wrote a function to poll the inputs of a data acquisition card at a set interval, your task would have less than favorable results. Such a system cannot guarantee the scheduling priority that it would receive. In the standard Linux operating system, as illustrated in Figure 2, all of the system processes are isolated from the hardware. This would not be so bad if our data polling was the only process the system performed. Our project, however, required user-space programs and a guarantee that the sensor inputs were polled every 1 millisecond. The hard real-time system would guarantee that sensor inputs would not be missed.

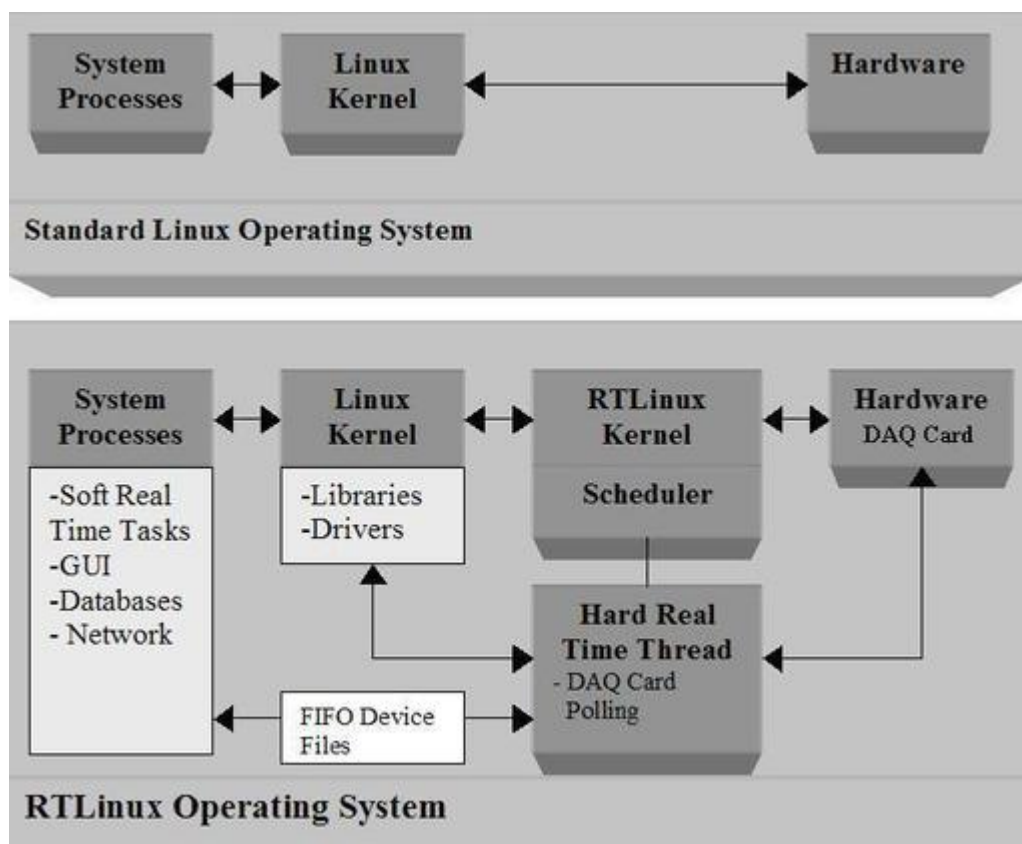


Figure 2. RTLinux System Flow

In the RTLinux operating system, also illustrated in Figure 2, the real-time task is isolated from the rest of the system processes and is implemented as a module. The module gains direct access to the hardware and the DAQ card drivers while receiving the priority that it needs over the rest of the system. The module is written to perform a specific task that produces reliable results and presents them to the user through the FIFO device files. The code for the module is kept simple, performing only tasks that must be held to hard real-time restraints. Connecting a handler function to one of the FIFO device files can control the module from the operator interface program. This structure, produced by RTLinux, ensures that the kernel cannot delay important module task with secondary priorities.

Data Acquisition Threads

We needed to monitor only digital input status with hard real-time requirements, which made the real-time function fairly small. Polling the data inputs was made easy, because United Electronics Industries provided RTLinux drivers with their DAQ card. A DAQ card from ADLINK Technology, Inc., also was tested with drivers for the RTLinux platform, and it configured easily. Not many companies provided such a service. The Comedi Project, however, offers another option for open-source drivers, tools and libraries for data acquisition.

The real-time task was written in the form of a loadable module, which has to have at least two functions: `init_module`, called when the module is inserted into the kernel, and `cleanup_module`, which is called right before it was removed (Listing 1).

Once the base module structure was established, we needed to create a thread for our real-time task in the module. The thread was created inside of the `init_module` and was set up to run with established RTLinux priorities. Establishing the priority and rate of execution for the thread was an important step to creating our real-time task.

FIFO Device Files

With a task running with predictable timing, real-time memory for data transfer was needed. The user-level task needed to access collected data and to control the real-time task. Real-time FIFOs are queues that can be read from and written to by Linux processes. The real-time FIFO devices are built during the RTLinux installation and created in the initialization of the real-time modules. Now a handler can be created and tied to one of the FIFO Devices. The handler can be set up to execute when 1 is written to the handler FIFO from the user interface program as controlling of the module is needed.

The module was created in order to be installed into the kernel as a real-time task. The tricky part of the real-time module was setting up the framework. Our real-time task code was straightforward, but lengthy. So, we've included only the real-time module skeleton (Listing 1). Notice the simplicity of the code that is implemented with real-time requirements.

Listing 1. Example Real-Time Module Code Skeleton

```
#include <pthread.h>
#include <rtl_fifo.h>
#include <rtl_core.h>
#include <rtl_time.h>
#include <rtl.h>
#include <rtl_fifo.h>

pthread_t thread_variable;

void thread_name(void)
{
    Struct Sched_param p;
    p.sched_priority = 1
    pthread_setschedparam(pthread_self(),
                          SCHED_FIFO, &p);
    pthread_make_periodicnp(pthread_self(),
                           getrttime(), 1000000);
    while(1) {
        // Real Time Task Code
        // Poll Data input lines, count low to high
        // transitions
        rtf_put() // Counts to be transferred by FIFO
        pthread_wait_np();
    }
}

int handler_function(){
    // Code tied to the handler FIFO
    // Variables for counting above are cleared out
}

int init_module(void)
{
    ififo_status = rtl_create(unsigned int fifo,
                             int size)
    pthread_create(&thread_variable, NULL,
                  thread_name, NULL);
    rtf_create_handler(FIFO_Number,
                      &handler_function)
}

int cleanup_module(void)
{
    rtf_destroy(unsigned int fifo)
    pthread_cancel(thrad_variable)
}
```

User Interface

With the real-time task up and running, the next step was to build the user interface for the project. The GUI was one of many tasks in our program that did not need to be accomplished within hard real-time restraints. We did not have heavy concerns for system resources, because our real-time module would execute within our established hard real-time requirements. We selected KDevelop IDE and Qt Designer GUI builder from Trolltech for programming the

interface. Qt Designer allowed for the development of a GUI with signal and slot interaction with functions in the KDevelop program. The result of developing with the combination of the two packages was perfect for our application. We were able to develop a user-friendly interface quickly.

The program was written to utilize two forms of information feed to the system: the digital inputs from the DAQ card and the human operator interaction with the GUI. The combination of the two was needed to maintain the integrity of the data. For example, the operator would enter the part number for the job being manufactured. The data that was gathered during the run would be tied to the part number. This replaced the old method of having the user tally information on handwritten forms.

Managing the Data

Gathering the data was only the first step for the system and program. The most important aspect was to make it usable by all of the departments of the company. The scheduling department, material purchasing department and tool service departments are some examples of where the data will be put to good use.

The SmartPress program stored the gathered data in a PostgreSQL database. PostgreSQL also is our enterprise back-end database. So, future application development will make SmartPress data visible in enterprise applications across the company.

Summary—Do-It-Yourself Information Technology

Our SmartPress system has moved from the test bed to the production floor. Looking back, we have created a flexible manufacturing monitoring system. Using Linux, the result is an expandable system that has been customized to the needs of the company. We can adapt the SmartPress for new manufacturing processes. The system also is upgradeable. In the future, we intend to rebuild the project on a newer kernel. For the upgrade, we probably will use Fedora Core as the base Linux platform.

The SmartPress system's low hardware cost is important as we are installing a system for each stamping press line. We cut hardware costs by using personal computers and DAQ boards that were supported under Linux. Our software development also was inexpensive. The time that Ryan Walsh spent on this project was similar to learning and developing in a proprietary controls language. Now, Ryan is fluent with kernel modules, real-time operating systems, PostgreSQL and GUI development. These skills are much more useful than learning one vendor's control programming language. For us, the do-it-

yourself option resulted in lower cost, no vendor tie-in and upgraded developer skills.

Resources for this article: www.linuxjournal.com/article/7810.

Craig Swanson (craig.swanson@slssolutions.net) designs networks and offers Linux consulting at SLS Solutions. He also develops Linux software at Midwest Tool & Die. Craig has used Linux since 1993.

Ryan Walsh (ryan.walsh@midwest-tool.com) works as a Network Engineer at Midwest Tool & Die. Ryan spends his spare time in free fall, jumping out of airplanes.

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

[Advanced search](#)

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

[Advanced search](#)

At the Forge

Aggregating Syndication Feeds

Reuven M. Lerner

Issue #128, December 2004

So far, we have looked at ways in which people might create RSS and Atom feeds for a Web site. Of course, creating syndication feeds is only one half of the equation. Equally as important and perhaps even more useful is understanding how we can retrieve and use syndication feeds, both from our own sites and from other sites of interest.

Over the last few months, we have looked at RSS and Atom, two XML-based file formats that make it easy to create and distribute summaries of a Web site. Although such syndication, as it is known, traditionally is associated with Weblogs and news sites, there is growing interest in its potential for other uses. Any Web-based information source is a potentially interesting and useful candidate for either RSS or Atom.

So far, we have looked at ways in which people might create RSS and Atom feeds for a Web site. Of course, creating syndication feeds is only one half of the equation. Equally as important and perhaps even more useful is understanding how we can retrieve and use syndication feeds, both from our own sites and from other sites of interest.

As we have seen, three different types of syndication feeds exist: RSS 0.9x and its more modern version, RSS 2.0; the incompatible RSS 1.0; and Atom. Each does roughly the same thing, and there is a fair amount of overlap among these standards. But networking protocols do not work well when we assume that everything is good enough or close enough, and syndication is no exception. If we want to read all of the syndicated sites, then we need to understand all of the different protocols, as well as versions of those protocols. For example, there actually are nine different versions of RSS, which when combined with Atom, brings us to a total of ten different syndication formats that a site might be using. Most of the differences probably are negligible, but it

would be foolish to ignore them completely or to assume that everyone is using the latest version. Ideally, we would have a module or tool that allows us to retrieve feeds from a variety of different protocols, papering over the differences as much as possible while still taking advantage of each protocol's individual power.

This month, we look at the Universal Feed Parser, an open-source solution to this problem written by Mark Pilgrim. Pilgrim is a well-known Weblog author and Python programmer, and he also was one of the key people involved in the creation of the Atom syndication format. This should come as no surprise, given the pain that he experienced in writing the Universal Feed Parser. It also handles CDF, a proprietary Microsoft format used for the publication of such items as active desktop and software updates. This part might not be of interest to Linux desktop users, but it raises interesting possibilities for organizations with Microsoft systems installed. The Universal Feed Parser (feedparser), in version 3.3 as of this writing, appears to be the best tool of its kind, in any language, and regardless of licensing.

Installing feedparser

Installing feedparser is extremely simple. Download the latest version, move into its distribution directory and type `python setup.py install`. This activates Python's standard installation utility, placing the feedparser in your Python site-packages directory. Once you have done installed feedparser, you can test it using Python interactively, from a shell window:

```
>>> import feedparser
```

The `>>>` symbols are Python's standard prompt when working in interactive mode. The above imports the feedparser module into Python. If you have not installed feedparser, or if something went wrong with the installation, executing this command results in a Python ImportError.

Now that we have imported our module into memory, let's use it to look at the latest news from *Linux Journal's* Web site. We type:

```
>>> ljfeed = feedparser.parse  
↳("http://www.linuxjournal.com/news.rss")
```

We do not have to indicate the protocol or version of the feed we are asking feedparser to work with—the package is smart enough to determine such versioning on its own, even when the RSS feed fails to identify its version. At the time of writing, the *LJ* site is powered by PHPNuke and the feed is identified explicitly as RSS 0.91.

Now that we have retrieved a new feed, we can find out exactly how many entries we received, which largely is determined by the configuration of the server:

```
>>> len(ljfeed.entries)
```

Of course, the number of items is less interesting than the items themselves, which we can see with a simple for loop:

```
>>> for entry in ljfeed.entries:
...     print entry['title']
... 
```

Remember to indent the print statement to tell Python that it's part of the loop. If you are new to Python, you might be surprised by the lines that begin with ... and indicate that Python is ready and waiting for input after the for. Simply press <Enter> to conclude the block begun by for, and you can see the latest titles.

We also can get fancy, looking at a combination of URL and title, using Python's string interpolation:

```
>>> for entry in ljfeed.entries:
...     print '<a href="%s">%s</a>' % \
...         (entry['link'], entry['title'])
```

As I indicated above, feedparser tries to paper over the differences between different protocols, allowing us to work with all syndicated content as if it were roughly equivalent. I thus can repeat the above commands with the syndication feed from my Weblog. I recently moved to WordPress, which provides an Atom feed:

```
>>> altneufeed = feedparser.parse(
...     "http://altneuland.lerner.co.il/wp-atom.php")
>>> for entry in altneufeed.entries:
...     print '<a href="%s">%s</a>' % \
...         (entry.link, entry.title)
```

Notice how this last example uses attributes `entry.link` and `entry.title`, while the previous example uses dictionary keys `entry['link']` and `entry['title']`. `feedparser` tries to be flexible, providing several interfaces to the same information to suit different needs and styles.

How New Is that News?

The point of a news aggregator or other application that uses RSS and Atom is to collect and present newly updated information. An aggregator can show only

the items that a server provides; if an RSS feed includes only the two most recently published items, then it becomes the aggregator's responsibility to poll, cache and display those items no longer being syndicated and summarized.

This raises two different but related questions: How can we ensure that our aggregator displays only items we have not seen yet? And is there a way for our aggregator to reduce the load on Weblog servers, retrieving only those items that were published since our last visit? Answering the first question requires looking at the modification date, if it exists, for each item.

The latter question has, as of this writing, been an increasingly popular issue of debate in the Web community. As a Weblog grows in popularity, the number of people who subscribe to its syndication feed also grows. If a Weblog has 500 subscribers to its syndication feed, and if each of these subscribers' aggregators look for updates each hour, that means an additional 500 requests per hour are made against a Web server. If the syndication feed provides the site's entire content, this can result in a great deal of wasted bandwidth—reducing the site's response time for other visitors and potentially forcing the site owner to pay for exceeding allocated monthly bandwidth.

feedparser allows us to be kind to syndicating servers and ourselves by providing a mechanism for retrieving a syndication feed only when there is something new to show. This is possible because modern versions of HTTP allow the requesting client to include an If-Modified-Since header, followed by a date. If the requested URL has changed since the date mentioned in the request, the server responds with the URL's content. But if the requested URL is unchanged, the server returns a 304 response code, indicating that the previously downloaded version remains the most current content.

We accomplish this by passing an optional modified parameter to our call to `feedparser.parse()`. This parameter is a standard, as defined by the time module, Python tuple in which the first six elements are the year, month number, day number, hour, minutes and seconds. The final three items don't concern us, and can be left as zeroes. So if I were interested in seeing feeds posted since September 1, 2004, I could say:

```
last_retrieval = (2004, 9, 1, 0, 0, 0, 0, 0, 0)
ljfeed = feedparser.parse(
    "http://www.linuxjournal.com/news.rss")
```

If *Linux Journal's* server is configured well, the above code either results in `ljfeed` containing the complete syndication feed—returned with an HTTP OK status message, with a numeric code of 200—or an indication that the feed has not changed since its last retrieval, with a numeric code of 304. Although

keeping track of the last time you requested a particular syndication feed might require more record-keeping on your part, it is important to do, especially if you request feed updates on a regular basis. Otherwise, you might find your application unwelcome at certain sites.

Working with Feeds

Now that we have a basic idea of how to work with feedparser, let's create a simple aggregation tool. This tool gets its input from a file called feeds.txt and produces its output in the form of an HTML file called feeds.html. Running this program by cron and looking at the resulting HTML file once per day provides a crude-but-working news feed from the sites that most interest you.

Feeds.txt contains URLs of actual feeds rather than of the sites from which we would like to get the feed. In other words, it's up to the user to find and enter the URL for each feed. More sophisticated aggregation tools usually are able to determine the feed's URL from a link tag in the header of the site's home page.

Also, despite my above warning that every news aggregator should keep track of its most recent request so as not to overwhelm servers, this program leaves out such features as part of my attempt to keep it small and readable.

The program, aggregator.py, can be read in Listing 1 and is divided into four parts:

1. We first open the output file, which is an HTML-formatted text file called myfeeds.html. The file is designed to be used from within a Web browser. If you are so inclined, you could add this local file, which has a file:/// URL, to your list of personal bookmarks or even make it your startup page. After making sure that we indeed can write to this file, we start the HTML file.
2. We then read the contents of feeds.txt, which contains one feed URL per line. In order to avoid problems with whitespace or blank lines, we strip off the whitespace and ignore any line without at least one printable character.
3. Next, we iterate over the list of feeds, feeds_list, invoking feedparser.parse() on that URL. When we receive a response, we write it to the output file, myfeeds.html, with both the URL and the title of the article.
4. Finally, we close the HTML and the file.

Listing 1. aggregator.py

```
#!/usr/bin/python
```

```

import feedparser
import sys

# -----
# Open the personal feeds output file

aggregation_filename = "myfeeds.html"
max_title_chars = 60

try:
    aggregation_file = open(aggregation_filename, "w")
    aggregation_file.write("""<html>
<head><title>My news</title></head>
<body>""")
except IOError:
    print "Error: cannot write '%s' " % \
        aggregation_filename
    exit

# -----
# Each non-blank line in feeds.txt is a feed source.

feeds_filename = "feeds.txt"
feeds_list = []

try:
    feeds_file = open(feeds_filename, 'r')
    for line in feeds_file:
        stripped_line = line.strip().rstrip()

        if len(stripped_line) > 0:
            feeds_list.append(stripped_line)
            sys.stderr.write("Adding feed '" + \
                stripped_line + "'\n")

    feeds_file.close()

except IOError:
    print "Error: cannot read '%s' " % feeds_filename
    exit

# -----
# Iterate over feeds_list, grabbing the feed for each

for feed_url in feeds_list:
    sys.stderr.write("Checking '%s'..." % feed_url)
    feed = feedparser.parse(feed_url)
    sys.stderr.write("done.\n")

    aggregation_file.write('<h2>%s</h2>\n' % \
        feed.entries[0].title)

    # Iterate over each entry from this feed,
    # displaying it and putting it in the summary
    for entry in feed.entries:
        sys.stderr.write("\tWrote: '%s'" % \
            entry.title[0:max_title_chars])

        if len(entry.title) > max_title_chars:
            sys.stderr.write("...")

        sys.stderr.write("\n")

        aggregation_file.write(
            '<li><a href="%s">%s</a>\n' %
            (entry.link, entry.title))

    aggregation_file.write('</u2>\n')

# -----
# Finish up with the HTML

aggregation_file.write("""</body>
</html>
""")
aggregation_file.close()

```

As you can see from looking at the code listing, creating such a news aggregator for personal use is fairly simple and straightforward. This is merely a skeletal application, however. To be more useful in the real world, we probably would want to move feeds.txt and myfeeds.html into a relational database, determine the feed URL automatically or semi-automatically based on a site URL and handle categories of feeds, so that multiple feeds can be read as if they were one.

If the above description sounds familiar, then you might be a user of Bloglines.com, a Web-based blog aggregator that probably works in the above way. Obviously, Bloglines handles many more feeds and many more users than we had in this simple toy example. But, if you are interested in creating an internal version of Bloglines for your organization, the combination of the Universal Feed Parser with a relational database, such as PostgreSQL, and some personalization code is both easy to implement and quite useful.

Conclusion

The tendency to reinvent the wheel often is cited as a widespread problem in the computer industry. Mark Pilgrim's Universal Feed Parser might fill only a small need in the world of software, but that need is almost certain to grow as the use of syndication increases for individuals and organizations alike. The bottom line is if you are interested in reading and parsing syndication feeds, you should be using feedparser. It is heavily tested and documented, often updated and improved and it does its job quickly and well.

Reuven M. Lerner, a longtime Web/database consultant and developer, now is a graduate student in the Learning Sciences program at Northwestern University. His Weblog is at altneuland.lerner.co.il, and you can reach him at reuven@lerner.co.il.

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

[Advanced search](#)

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

[Advanced search](#)

Kernel Korner

Unionfs: Bringing Filesystems Together

Charles P. Wright

Erez Zadok

Issue #128, December 2004

Unionfs merges several directories into a single unified view. We describe applications of Unionfs and also interesting implementation aspects.

For ease of management, it can be useful to keep related but different sets of files in separate locations. Users, however, often prefer to see these related files together. In this situation, unioning allows administrators to keep such files separate physically, but to merge them logically into a single view. A collection of merged directories is called a union, and each physical directory is called a branch. As shown in Figure 1, Unionfs simultaneously layers on top of several filesystems or on different directories within the same filesystem. This layering technique is known as stacking (see the on-line Resources for more on stacking). Unionfs presents a filesystem interface to the kernel, and in turn Unionfs presents itself as the kernel's VFS to the filesystems on which it stacks. Because Unionfs presents a filesystem view to the kernel, it can be employed by any user-level application or from the kernel by the NFS server. Because Unionfs intercepts operations bound for lower-level filesystems, it can modify operations to present the unified view. Unlike earlier stackable filesystems, Unionfs is a true fan-out filesystem; it can access many underlying branches directly.

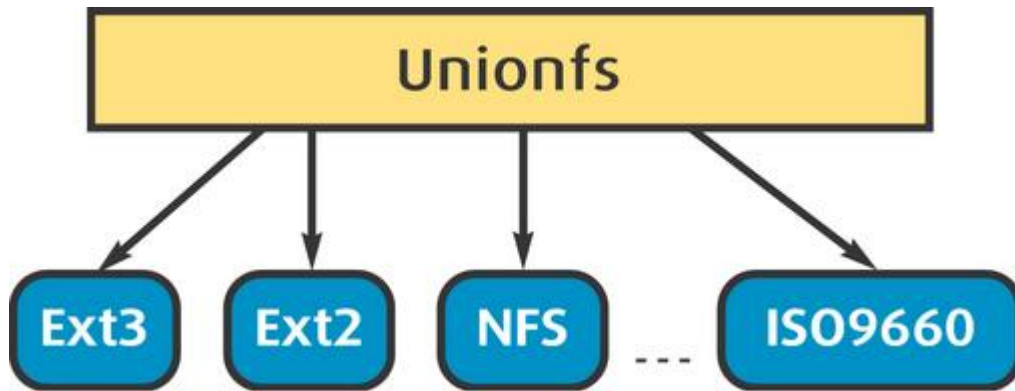


Figure 1. A union consists of several underlying branches, which can be of any filesystem type.

Unionfs Semantics and Usage

In Unionfs, each branch is assigned a precedence. A branch with a higher precedence overrides a branch with a lower precedence. Unionfs operates on directories. If a directory exists in two underlying branches, the contents and attributes of the Unionfs directory are the combination of the two lower directories. Unionfs automatically removes any duplicate directory entries, so users are not confused by duplicated filenames or directories. If a file exists in two branches, the contents and attributes of the Unionfs file are the same as the file in the higher-priority branch, and the file in the lower-priority branch is ignored.

As a concrete example, assume that we unify two directories, /Fruits and /Vegetables:

```

$ ls /Fruits
Apple Tomato
$ ls /Vegetables
Carrots Tomato
$ cat /Fruits/Tomato
I am botanically a fruit.
$ cat /Vegetables/Tomato
I am horticulturally a vegetable.
  
```

To use Unionfs, you first need to compile the Unionfs module and load it into the kernel. Next, like any other filesystem, Unionfs is mounted. Unlike other filesystems, Unionfs does not mount on top of a device; it mounts on top of directories that are specified as a mount-time option. To create a union, we mount Unionfs as follows:

```

# mount -t unionfs -o dirs=/Fruits:/Vegetables \
> none /mnt/healthy
  
```

In this example, the mount option `dirs` tells Unionfs which directories make up the union. Unionfs does not mount any device, so we use `none` as a

placeholder. Finally, /mnt/healthy is the location of the merged view. Now /mnt/healthy contains three files: Apple, Carrots and Tomato. Because we specified /Fruits before /Vegetables, /mnt/healthy/Tomato contains "I am botanically a fruit." If we were to reverse the dirs= option, /mnt/healthy/Tomato would contain "I am horticulturally a vegetable." (which agrees with the 1893 U.S. Supreme Court ruling on the matter).

This process is recursive. If there were a subdirectory of Fruits named Green that contained a file named Lime and a subdirectory of Vegetables also named Green that contained a file named Lettuce, the result would be:

```
$ ls /mnt/healthy
Apple Carrots Green/ Tomato
$ ls /mnt/healthy/Green
Lime Lettuce
```

Unionfs can be applied in several ways. Simple examples include unifying home directories from multiple servers or merging split ISO images to create a unified view of a distribution. In a similar vein, Unionfs, with copy-on-write semantics, can be used to patch CD-ROMs, for source code management or for snapshotting.

Unified Home Directories

Often a single client machine mounts home directories from several different NFS servers. Unfortunately, each server has a distinct mountpoint, which is inconvenient for users. It would be ideal if all home directories were available from the same place (/home for example). Some automounters use symbolic links to create the illusion of a union. With Unionfs, these links are not necessary. The separate exported directories simply can be unified into a single view. Assume we have two filesystems, one mounted on /alcid and the other mounted on /penguin. We can unify them into /home as follows:

```
# mount -t unionfs -o dirs=/alcid,/penguin \
> none /home
```

Now home directories from both /alcid and /penguin are available from /home.

Unionfs supports multiple read-write branches, so the user's files will not migrate from one directory to another. This contrasts with previous unioning systems, such as BSD-4.4's Union Mounts, which generally supported only a single read-write branch.

Merging Distribution ISO Images

Most Linux distributions are available as both ISO images and also as individual packages. ISO images are convenient because they can be burnt directly to CD-ROMs, and you need to download and store only a few files. To install to a machine over a network, however, you often need to have the individual packages in a single directory. Using the loopback device, the ISO images can be mounted on separate directories, but this layout is not suitable for a network install because all files need to be located in a single tree. For this reason, many sites maintain copies of both the ISO images and also the individual package files, wasting both disk space and bandwidth and increasing management efforts. Unionfs can alleviate this problem by virtually combining the individual package directories from the ISO images.

In this example, we are mounting over two directories, `/mnt/disc1` and `/mnt/disc2`. The mount command is as follows:

```
# mount -t unionfs -o dirs=/mnt/disc1,/mnt/disc2 \  
> none /mnt/merged-distribution
```

Now `/mnt/merged-distribution` can be exported using NFS or FTP for use in network installs.

Copy-on-Write Unions

In the previous example of the ISO images, all of the branches in the union were read-only; hence, the union itself was read-only. Unionfs also can mix read-only and read-write branches. In this case, the union as a whole is read-write, and Unionfs uses copy-on-write semantics to give the illusion that you can modify files and directories on read-only branches. This could be used to patch a CD-ROM. If the CD-ROM is mounted on `/mnt/cdrom`, and an empty directory is created in `/tmp/cdpatch`, then Unionfs can be mounted as follows:

```
# mount -t unionfs -o dirs=/tmp/cdpatch,/mnt/cdrom \  
> none /mnt/patched-cdrom
```

When viewed through `/mnt/patched-cdrom`, it appears as though you can write to the CD-ROM, but all writes actually will take place in `/tmp/cdpatch`. Writing to read-only branches results in an operation called a copyup. When a read-only file is opened for writing, the file is copied over to a higher-priority branch. If required, Unionfs automatically creates any needed parent directory hierarchy.

In this CD-ROM example, the lower-level filesystem enforces the read-only permissions, and Unionfs respects them. In other situations, the lower-level filesystem may indeed be read-write, but Unionfs should not modify the branch. For example, you may have one branch that contains pristine kernel sources and then use a separate branch for your local changes. Through Unionfs, the pristine sources should be read-only, as the CD-ROM was in the previous example. This can be accomplished by adding `=ro` to a directory in the `dirs` mount option. Assume that `/home/cpw/linux` is empty, and `/usr/src/linux` contains a Linux kernel source tree. The following mount command makes Unionfs behave as a source code versioning system:

```
# mount -t unionfs -o \  
> dirs=/home/cpw/linux:/usr/src/linux=ro \  
> none /home/cpw/linux-src
```

This example makes it appear as if an entire Linux source tree exists in `/home/cpw/linux-src`, but any changes to that source tree, such as changed source files or new object files, actually go to `/home/cpw/linux`.

With a simple modification, we also can use an overlay mount. That is, we can replace `/home/cpw/linux` with the unified view:

```
# mount -t unionfs -o \  
> dirs=/home/cpw/linux:/usr/src/linux=ro \  
> none /home/cpw/linux
```

Implementation

Most filesystem operations in Unionfs move from higher-priority branches to lower-priority branches. For example, LOOKUP begins in the highest-priority branch in which the parent exists and then moves to lower-priority branches. During the lookup operation, Unionfs caches information for use in later operations.

CREATE attempts to create files in the highest-priority branch where the parent directory exists. The CREATE operation uses the cached lookup information to operate directly on the appropriate branch, so in effect, it moves from higher-priority branches to lower-priority branches.

Unionfs uses several techniques to provide the illusion of modifying read-only branches, and at the same time, maintains normal UNIX semantics. If there is an error while creating a file, error handling must be performed. Error handling proceeds from lower-priority branches to higher-priority branches. Starting in the highest-priority branch in which the parent exists, Unionfs attempts to

create the file in each higher-priority branch. Finally, if the operation fails in the highest-priority branch for the whole union, then Unionfs returns an error to the user.

In contrast to CREATE, the UNLINK operation always proceeds from lower-priority branches to higher-priority branches. Because the last underlying object to be UNLINKed is the highest-priority object, the user-visible state is not modified until the very end of Unionfs's UNLINK operation. The most complex situations to handle are partial errors. If removing an intermediate file fails and Unionfs simply removes the highest-priority file, a lower-priority file becomes visible to the user. To handle these error conditions, Unionfs uses a special high-priority file called a whiteout. If Unionfs encounters a whiteout, it behaves as if the file does not exist in any lower-priority branch. Internally, to create a whiteout for a file named F, Unionfs creates a zero-length file named .wh.F. Getting back to UNLINK—if an intermediate UNLINK has failed, instead of deleting the highest-priority file, Unionfs renames the file to the corresponding whiteout name.

This careful ordering of operations has two effects. The first is that UNIX semantics are maintained even in the face of errors or read-only branches. The user-visible state isn't modified until the operation is attempted on the highest-priority branch. The success or failure of the operation is determined by the success or failure of this branch. Through the use of whiteouts, a file can be deleted even if it exists on a read-only branch. The second effect is that when no errors occur, files and directories tend to stay in the branches where they were originally. This is important because one of the goals of Unionfs is to keep the files in separate places.

File Deletion Semantics

By default, Unionfs attempts to delete all instances of a file (or directory) in all branches; this mode is called DELETE_ALL. Aside from DELETE_ALL, Unionfs also supports two more deletion modes, DELETE_WHITEOUT and DELETE_FIRST. DELETE_WHITEOUT behaves like the default mode externally, but instead of removing all files in the Union, a whiteout is created. This has the advantage that the lower-priority files still are available through the underlying filesystem. DELETE_FIRST departs from classical UNIX semantics. It only removes the highest-priority entry in the union and, thus, allows the lower-priority entries to show through. These modes also are used for the RENAME operation, as it is a combination of a create followed by a delete.

DELETE_FIRST requires some user knowledge of the union's components. This is useful when Unionfs is used for source code versioning, as in our previous example of a kernel source tree. If we change a file in /home/cpw/linux, the file is copied up to the higher-priority branch. If the file is deleted with standard

DELETE_ALL semantics, Unionfs creates a whiteout in the highest-priority branch (because it cannot modify the read-only lower-priority branch). The original source file in the lower-priority branch is now inaccessible, so it must be copied into the union from the source, which hardly makes for a convenient versioning system. This situation is precisely where DELETE_FIRST comes in handy. The delete mode is specified as a mount option, as in the following example:

```
# mount -t unionfs -o \  
> dirs=/home/cpw/linux:/usr/src/linux=ro,\  
> delete=first none /home/cpw/linux
```

Now, as before, if we change a file in /home/cpw/linux, the changes don't affect /usr/src/linux. If we decide we don't like the changes, we simply can remove the file and the original version will show through.

Unionfs Snapshots

With the unionctl utility, Unionfs's branch configuration can change on the fly. New branches can be added anywhere in the union, branches can be removed and read-write branches can be marked read-only (or vice versa). This flexibility allows Unionfs to create filesystem snapshots. In this example, we use Unionfs to create a snapshot of /usr while installing a new package:

```
# mount -t unionfs -o dirs=/usr none /usr
```

At this point, Unionfs has a single branch that is read-write, /usr. All operations are passed to the lower-level filesystem, and it is as if Unionfs didn't exist.

Creating a snapshot involves two steps. The first is to specify the location of the snapshot files by adding a branch (say, /snaps/0), as follows:

```
# unionctl /usr --add /snaps/0
```

At this point, Unionfs creates new files for /usr in /snaps/0, but files in subdirectories of /usr are created in the underlying /usr. The reason for this seeming contradiction is the rule that files are created in the highest-priority branch where the parent exists. For files in the root directory of the union, /usr, the parent exists in both branches. Because /snaps/0 is the higher-priority branch, new files and directories are created physically in /snaps/0. However, /snaps/0 is empty, so if a file were created in /usr/local, the highest-priority parent actually would be in the underlying /usr branch.

To complete the migration, the original `/usr` branch needs to be read-only. Again, we use `unionctl` to modify the branch configuration:

```
# unionctl /usr --mode /usr ro
```

Now, because Unionfs thinks the underlying `/usr` is read-only, all write operations really take place in `/snaps/0`. Multiple snapshots can be taken simply by adding another branch, say, `/snaps/1`, and marking `/snaps/0` as read-only.

The first snapshot can be viewed through the underlying directory, `/usr`. Each snapshot consists of a base directory and several directories that have incremental differences. To view a specific snapshot, all we need is to unify the first snapshot and the incremental changes. For example, to view the snapshot that consists of `/usr` and `/snaps/0`, mount Unionfs as follows:

```
# mount -t unionfs -o ro,dirs=/snaps/0:/usr \  
> none /mnt/snap
```

In this example, Unionfs itself is mounted read-only, so the underlying directories are not modified.

After determining that the changes made in a snapshot are good, the next step often is to merge the snapshot back into the base. The Unionfs distribution includes a `snapmerge` script that applies incremental Unionfs snapshots to a base directory. This is done by recursively copying the files in the snapshot directory to the base. After the copy procedure is done, new files and changed files are completed. The last step is to handle file deletions, which is done by creating the list of whiteouts and deleting the corresponding files. The whiteouts themselves also are removed so as not to clutter the tree.

Conclusion

Unionfs recursively merges several underlying directories or branches into a single virtual view. The efficient fan-out structure of Unionfs makes it suitable for many applications. Unionfs can be used to provide merged distribution ISOs, a single `/home` namespace and more. Unionfs's copy-on-write semantics make it useful for source code versioning, snapshotting and patching CD-ROMs. We benchmarked Unionfs's performance under Linux 2.4. For a compile benchmark with one to four branches, Unionfs overhead is only 12%. For an I/O intensive workload, the overhead ranges from 10% for a single branch to 12% for four branches.

Acknowledgements

Thanks to Puja Gupta, Harikesavan Krishnan, Mohammad Zubair and Jay Dave, who also were on the Unionfs development team. Special thanks go to Mohammad for helping to get the software for this article prepared for release.

Resources for this article: www.linuxjournal.com/article/7812.

Charles P. Wright (cwright@cs.stonybrook.edu) is a Computer Science PhD student at Stony Brook University. Charles conducts operating systems research with a focus on filesystems, security and extensibility. He also is active in the Linux Users Group at Stony Brook (LUGSB).

Erez Zadok (ezk@cs.stonybrook.edu) is on the Computer Science faculty at Stony Brook University, the author of *Linux NFS and Automounter Administration* (Sybex, 2001), the creator and maintainer of the FiST stackable templates system and the primary maintainer of the Am-utils (aka Amd) automounter. Erez conducts operating systems research with a focus on filesystems, security, versatility and portability.

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

[Advanced search](#)

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

[Advanced search](#)

Cooking with Linux

Lights... Camera... Action!

Marcel Gagné

Issue #128, December 2004

Set up a simple video studio with only a Webcam, microphone and these open-source packages.

I'm afraid it's true, François. The camera does not lie. That's what you look like when you are serving wine. Of course, *mon ami*, I have no intention of putting this little video on our restaurant's Web site. Why am I doing this? Ah, you have not yet looked at the feature for today's menu. Multimedia and entertainment is the ticket, *mon ami*, and you have to admit, this little video certainly is entertaining. Don't be upset, François, I have the ultimate respect for you. Besides, there's no time to fret, our guests will be here any moment. Ah, you see, they are already at the door.

Welcome, *mes amis*, to *Chez Marcel!* It's wonderful to see you all today here at the finest Linux restaurant on the planet, which also happens to be the home of one of the greatest wine cellars in the world. Please sit and make yourselves comfortable. François was just heading to the wine cellar now. The 2001 Santa Lucia Highlands Pinot Noir is drinking very nicely. It's a showy wine, bursting with hints of stardom—perfect for today's menu.

As you will notice, each workstation today is equipped with a small Webcam and microphone. The immediate idea is that we (or someone else) will wind up as the star of our little movie, but sometimes the software itself is the star. After all, many an application has shone on *Chez Marcel's* menu, *non?* And, this is where Karl Beckers' xvidcap comes into play. xvidcap is a great little tool for recording the action on your X desktop using a program called ffmpeg, which likely is already on your system. The result is an MPEG movie of your entire desktop or any portion thereof.

Why would you do such a thing, you might ask? Well, you can use this to create a training video to show others how to use an application or to show the world how good you are at playing your favorite first-person shooter. To get your copy of xvidcap, head on over to the official Web site (see the on-line Resources). The site offers both source and some binary packages, such as RPM or DEB. Should your particular distribution not be covered, building xvidcap from source is a simple extract and build five-step:

```
tar -xzvf xvidcap-1.1.3.tar.gz
cd xvidcap-1.1.3
./configure
make
su -c "make install"
```

The program name is either xvidcap or gvidcap. I say either because if you have the GTK2+ development libraries (version 2.2.1 or later), you get a second binary. Both work the same but, quite honestly, the GTK2+ interface looks and works better. If you don't have a recent GTK2+, you still may be able to get the interface by using one of the binary packages as opposed to source.

When you start either version of the program from the command line, you see a simple toolbar with a few buttons and a red rectangle floating below it. This is the capture window, and you can move it to cover whatever area you want to record. The default window, however, is fairly small. To change the area you want to record, click the crosshairs icon (second from the right on the toolbar), then click and drag the pointer to take in whatever area you want to include in your capture. The red rectangle will be resized. The toolbar contains buttons to record, stop, pause and so on. Pausing over the buttons will display tool tips. That's it, you are recording a movie of whatever transpires inside the red rectangle (Figure 1).

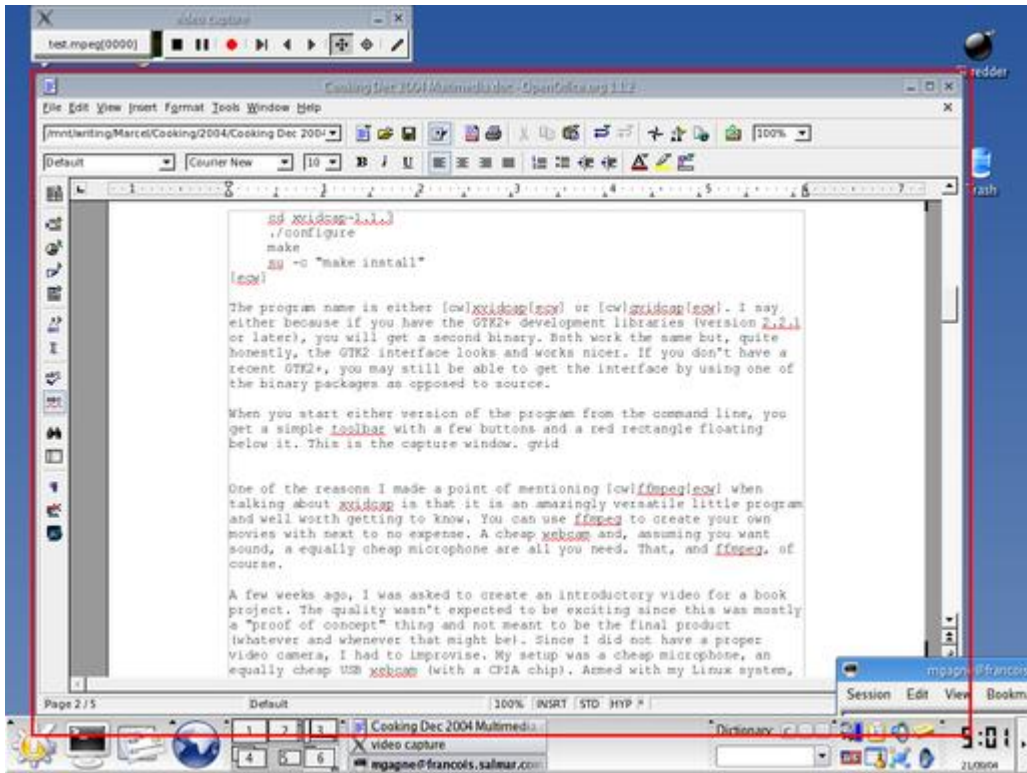


Figure 1. Selecting a Capture Window with xvidcap/gvidcap

Some preference settings let you change the number of frames per second, the video codec and whether you want to capture the mouse pointer in the final video. Right-click on the frame button (first on the left) and select Preferences (Figure 2). After you've experimented for a while, type `man xvidcap` for details on running the program without the GUI. The added bonus is that you can capture audio as well, which is perfect if you are creating an instructional video.

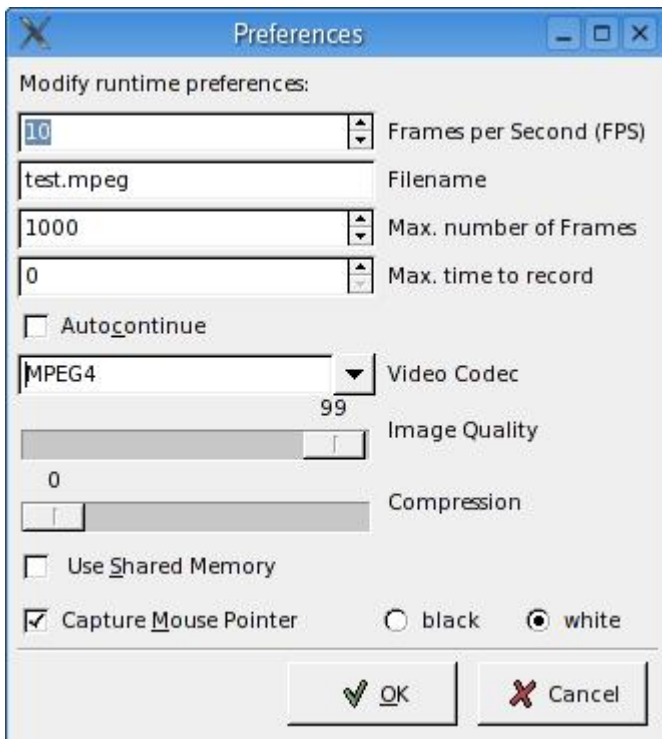


Figure 2. The gvidcap Preferences Dialog

One of the reasons I made a point of mentioning ffmpeg when introducing xvidcap is that it is an amazingly versatile little program and well worth getting to know. You can use ffmpeg to create your own movies with next to no expense. A cheap Webcam and, assuming you want sound, an equally cheap microphone are all you need (and ffmpeg, of course).

A few weeks ago, I was asked to create an introductory video for a book project. The quality wasn't expected to be exciting, because this was mostly a proof-of-concept thing and not meant to be the final product. Because I did not have a proper video camera, I had to improvise and thereby created what might well be the world's cheapest video recording studio (Figure 3). My setup was a cheap microphone and a cheap USB Webcam with a CPIA chip. Armed with my Linux system, I figured I was ready to go. The only question was "How?"



Figure 3. Is this the world's cheapest video recording studio?

Using ffmpeg, I created my video clip, experimented with timing, frame rate and so on, until I had something that was getting pretty good. With the following command, I created an AVI format video clip at ten frames per second:

```
ffmpeg -vd /dev/video -ad /dev/sound/dsp -r 10 \  
-s 352x288 video_test.avi
```

That's it. My video device input (indicated by the `-vd` option) is `/dev/video` and my sound, or audio, input was `/dev/sound/dsp`. That's the `-ad` option. Your own devices may be a little different of course. For instance, on another of my machines, the USB video device is `/dev/video0`. As it stands, the command continues to record until you run out of disk space. So, to limit a recording to ten seconds, I use the `-t` option:

```
ffmpeg -vd /dev/video -ad /dev/sound/dsp -r 10 \  
-s 352x288 -t 10 video_test.avi
```

The resulting clip, which I called `video_test.avi`, can be played with Xine, Kmpayer, MPlayer or whatever video player you prefer.

Well, *mes amis*, my foray into video production on the cheap was a success, or so I thought. When I sent the video clip for approval, I was asked whether they could have it in MPG format instead. Rather than redoing what obviously was a great work of cinematography, I converted it using the following command:

```
ffmpeg -i video_test.avi video_test.mpg
```

As versatile as `ffmpeg` is, you certainly should spend a few minutes looking at the man page. I promise you'll discover a lot of really cool uses for it.

For those of you who already own a digital video camera, recording high-quality video is not a problem. The challenge is to get the video on your hard disk for editing, compressing and eventually burning to disk for sending to friends and family. I have a Sony Handycam. It comes with a USB port, but it also has a much faster means of getting the video transferred through a high-performance serial bus or, as most people know it, a FireWire port. The real techies out there refer to it as an IEEE1394 port. Your computer also needs to have such a port (or card) in order to transfer the information, but the performance of the IEEE1394 port truly is worth the investment.

To manipulate the resulting video in a friendly way, however, you need to get your hands on a video editor like Arne Schirmacher's Kino (see Resources). The `dvgrab` program, also available from the Kino Web site, is built-in to Kino so you don't need a separate copy. You see, *mes amis*, like `xvidcap`, Kino uses a few command-line tools under the pretty interface. One of those tools is our old friend `ffmpeg`.

These days, Arne has some great developers working on the project, and they have put together a rather nice and easy-to-use video editor for Linux. See the

feature article on page XX. The beauty of Kino is that you can use it to extract and edit video directly from your IEEE1394-compatible video camera. If you have such a device, which can be connected with a FireWire cable, Kino even lets you control the device through the application. Before I continue on, let me step back and say something about the IEEE1394 support. Any recent Linux distribution should have IEEE1394 support included (through loadable kernel modules), but on my test system, the drivers weren't loaded automatically. No problem—I loaded them manually:

```
modprobe ohci1394
modprobe raw1394
chmod 666 /dev/raw1394
```

Getting your digital video across the IEEE1394 cable and onto your computer can be done easily from the command line with `dvgrab`. After all, that's what Kino does to capture video. Although you can type `dvgrab` and start capturing, the best way to do this is by using the `-i` option, meaning interactive. You then can control the camera and capture using simple single key presses.

```
dvgrab -i
Going interactive. Press '?' for help.
q=quit, p=play, c=capture, Esc=stop, h=reverse, j=backward scan,
k=pause, l=forward scan, a=rewind, z=fast forward, 0-9=trickplay,
<space>=play/pause
Capture Started
```

We become accustomed to working with graphical interfaces, but this is a lot easier and nicer to use than it sounds. The resulting video file on your system is called `dvgrab-XXX.avi` by default, the `XXX` being a three-digit extension.

Kino's interface is clean and easy to navigate (Figure 4). To the right of the main window, tabs indicate the various functions, from capturing to editing to exporting the finished product. The Edit tab is where most of the work happens (after the Capture, of course). This is where you cut or join scenes, or even insert additional files in to your video project. The Timeline tab breaks up the current scene (and video clip) into multiple frames, so you can jump to any part of the video without having to play the whole thing.

An FX tab lets you add various special effects to your videos. These include things like reverse video, sepia tones, mirror and kaleidoscope effects and so on. Audio effects include fade in, fade out and silence.

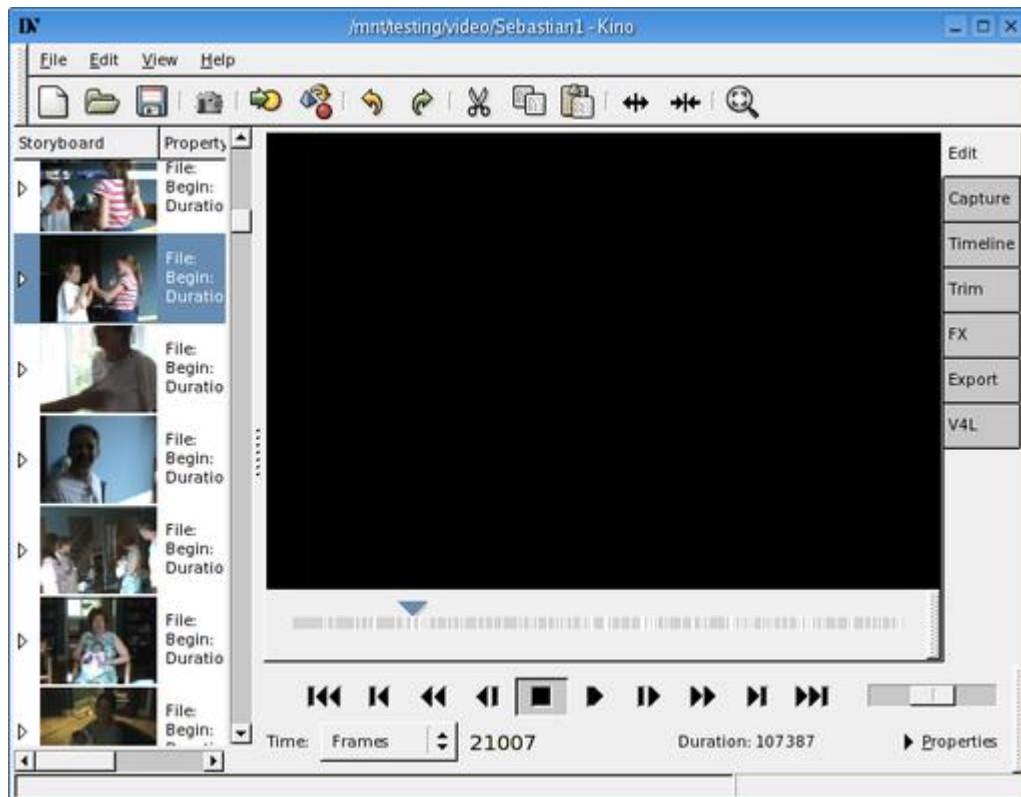


Figure 4. Editing a Home Movie with Kino

Once you start truly enjoying your digital video camera, you are going to record more than a few minutes of tape. The reason I mention this is that when you are capturing output from your digital camera with Kino, a number of large files will be created, each about 820MB in size and each sequential for the duration of the video. Those mega-files aren't what you want your final product to be, however. That's what the Export tab is all about. Various options are available here (in a group of sub-tabs) including exporting only the audio tracks to WAV files. I also like the fact that I can export stills of individual frames. The option I tend to use the most, however, is DV pipe, and this is where ffmpeg makes an encore appearance.

After all the editing, cutting and pasting is done, I export my finished product using ffmpeg to either video CD AVI files or DVD format. The resulting files (for example, VCD AVI format) are much smaller and much more manageable. One hour of digital video takes up about 9GB of disk space. The exported video, however, fits nicely on a single 700MB CD-ROM.

As you can see *mes amis*, your Linux system provides some great tools to make you, or your software, a star. Speaking of which, it seems as though closing time has arrived already and judging by the wine-induced smiles on some of your faces, we have the opportunity to capture some memorable video. I jest, of course. François, do be so kind as to refill our guests' glasses one more time. Perhaps what we should capture for posterity is our parting toast. Until next

time, *mes amis*, let us all drink to one another's health. *A votre santé Bon appétit!*

Resources for this article: www.linuxjournal.com/article/7808.

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of the all new *Moving to the Linux Business Desktop* (ISBN 0-131-42192-1), his third book from Addison-Wesley. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy, and folds a mean origami T-Rex.

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

[Advanced search](#)

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

[Advanced search](#)

Paranoid Penguin

Adding Clam Antivirus to Your Postfix Server

Mick Bauer

Issue #128, December 2004

Keep vulnerable systems out of trouble with a layer of high-performance protection on your Linux mail server.

The winner of *Linux Journal's* 2004 Editors' Choice Award for Security Tool was ClamAV, a 100% free and open-source virus scanner that runs on Linux but scans for viruses that affect a variety of platforms (see *Linux Journal*, August 2004). As Reuven Lerner noted in the award article, "ClamAV is giving the commercial virus-checking programs a real run for their money."

In this month's column, I show you how to harness the power of ClamAV on your Postfix e-mail gateway. Along the way, you also learn a few things about Amavisd-new, a powerful e-mail-processing daemon that serves as a crucial conduit between e-mail servers, such as Postfix and Sendmail, and mail-scanning tools, such as ClamAV and SpamAssassin.

The Architecture

The scenario I'm about to describe by no means represents the only good way to use ClamAV. But it's the scenario I personally have encountered the most; it's certainly typical. Say we have an SMTP gateway that receives all Internet e-mail destined for our organization, and we want to configure that SMTP gateway to pre-filter that mail for viruses (Figure 1). Our gateway can be configured to deliver mail to local mailboxes, or it can relay everything to an internal mail server. Everything that follows works the same regardless of the delivery method.

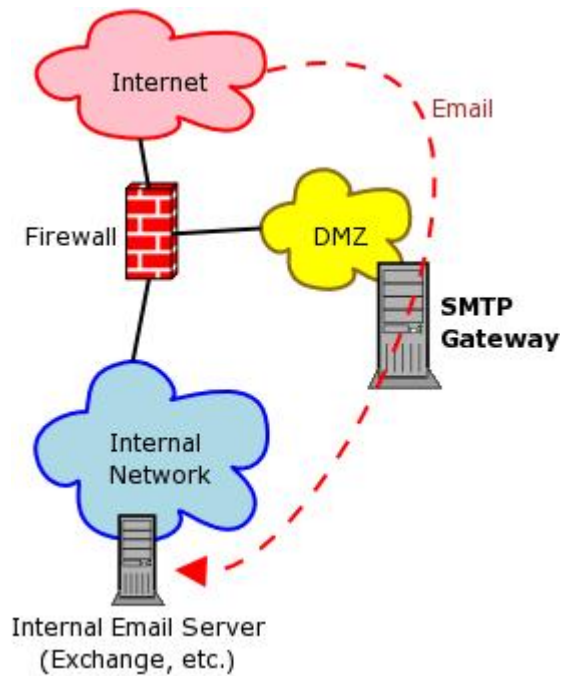


Figure 1. Our Example E-mail Architecture

In a high-volume setting, we could do all of our virus scanning on a standalone scanning server rather than locally on the SMTP gateway; all the tools described here work well that way. But for simplicity's sake and because it's common practice, we're going to run our virus scanner directly on the SMTP gateway.

We're going to use Postfix for our Mail Transfer Agent (MTA) because it is popular, securable and can work well with ClamAV. But Postfix can't interact directly with ClamAV, at least not reliably. ClamAV isn't too good yet at dissecting actual e-mail messages, as opposed to data streams. Therefore, we need to introduce a helper dæmon called Amavisd-new.

Amavisd-new is another free and open-source tool, and its sole purpose in life is to broker transactions between MTAs, such as Postfix and Sendmail, and anti-virus and anti-spam utilities, such as ClamAV and SpamAssassin. Among other things, Amavisd-new excels at converting MIME e-mail attachments into conventional data files that scanners can understand.

Amavisd-new's dæmon, amavisd, can communicate through a variety of protocols, including the SMTP and LMTP e-mail protocols, and also through UNIX sockets. Here, we configure amavisd to listen for e-mail by way of SMTP on TCP port 10024, communicate with ClamAV by using ClamAV's local UNIX socket and send e-mail and scanning-results back to Postfix on TCP port 10025. Figure 2 illustrates how e-mail flows through our SMTP gateway.

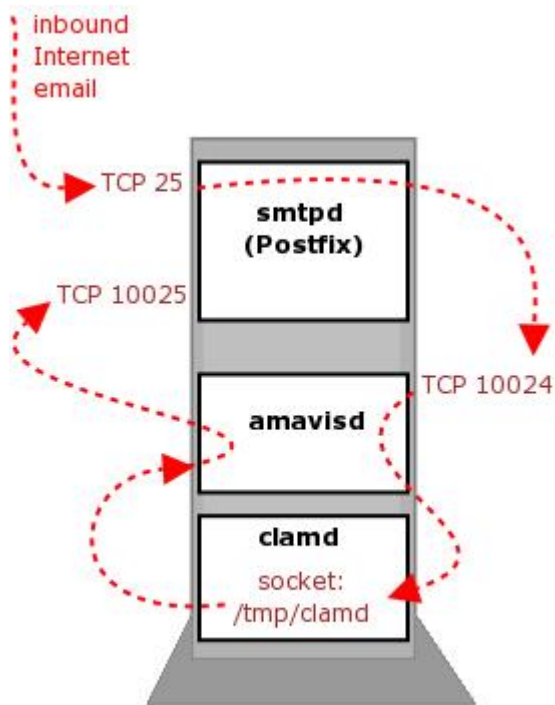


Figure 2. Postfix, Amavisd-new and ClamAV Data Flows

Getting and Installing the Software

Both ClamAV and Amavisd-new are written in Perl and depend on numerous Perl modules. Therefore, I recommend you find and use binary packages of recent versions of these two tools for your distribution. You should have a much easier time letting apt-get, Yum or up2date automatically handle the dependencies that arise when trying to install everything manually.

The ClamAV Web site, besides being the home of the latest ClamAV source code, has a page that lists sources of binary packages for ClamAV for a variety of Linux distributions and other operating systems. For Red Hat and Fedora users, Dag Wieers' page (see the on-line Resources) provides Yum repositories and up2date sources containing both ClamAV and Amavisd-new. The Amavisd-new Web site has links to other sources of Amavisd-new packages, as well as the latest Amavisd-new source code. ClamAV is now a standard package on Debian beginning with the sarge release, and Amavisd-new is part of SuSE 9.1.

If you install either package from source code or from a standalone package, as opposed to using Yum, up2date or apt-get, be sure to see the Prerequisites section of Amavisd-new's INSTALL instructions (see Resources). ClamAV's prerequisites aren't quite as well documented. When in doubt, it doesn't hurt to try `rpm --test -iv clamav_packagename.rpm` on your ClamAV RPM to see which required packages your system is lacking.

Chances are your distribution provides packages for the various Perl modules that ClamAV and Amavisd-new require. Any not provided can be obtained from CPAN or from other third-party sites that specialize in packaging software for your distribution.

Configuring ClamAV

Once you've installed ClamAV and Amavisd-new, you can configure them. We start with ClamAV, the simpler of the two. ClamAV's configuration file is `/etc/clamav.conf`. To configure ClamAV, open this file with the text editor of your choice. Listing 1 shows the parameters most people need to change from the defaults.

Listing 1. Non-Default Settings in `/etc/clamav.conf`

```
# Example
LogFile /var/log/clamd.log
LogFileMaxSize 5M
DatabaseDirectory /usr/share/clamav
LocalSocket /usr/share/clamav/clamd.sock
User clamav
```

The first line seems innocent enough, but it's important that you comment it out. If it isn't, clamd cannot run.

The two `LogFile...` settings are commented out by default. Uncomment them to enable logging to the file of your choice and to overwrite the file when it reaches the size specified in `LogFileMaxSize`.

`DatabaseDirectory` is crucial. This is where ClamAV keeps its virus-signature databases—its brains, as it were. In the ClamAV RPM I installed, the clamd daemon was compiled to use `/usr/share/clamav` for this setting, but the included sample `clamav.conf` file showed a commented-out value of `/var/lib/clamav`, so I uncommented this setting and changed it to `/usr/share/clamav` in order to minimize confusion.

`LocalSocket` determines which socket file clamd uses to communicate with the outside world, namely, Amavisd-new. If you use this setting, which I recommend, be sure to leave the lines for `TCPsocket` and `TCPAddr` commented out. On my Genco package, the default value for this path for `LocalSocket` is `/tmp/clamd`, but `/tmp` is world-readable/writable. I recommend you instead set this path to `/usr/share/clamav/clamd.sock` and set the permissions on `/usr/share/clamav` to `rwxrwx`. In other words, remove the read/write/execute bits for other.

The last setting in Listing 1, `User`, specifies the name of the non-privileged user account clamd should run as after initialization. clamd must be started as root,

but if this parameter is uncommented and set, clamd demotes itself after it's done initializing.

This is all most of us need to set in `/etc/clamav.conf`. Before you start clamd, however, make sure your system has an account for clamav and the permissions on any path you set in `/etc/clamav.conf` are set accordingly. It's also a good idea to use a group, clamav, for this account. As we see in the next section, this makes it easier to share certain resources between clamd and amavisd.

The `/etc/passwd` entry for clamav user account is:

```
clamav:x:52:52:ClamAV Daemon:/:/bin/false
```

And, the `/etc/group` entry for clamav group account is:

```
clamav:x:52:
```

Once clamd is configured, you can start it simply by entering the command `clamd`. If you installed ClamAV from a binary package, your system may have an init script for clamd in `/etc/init.d`. If so, be sure to enable it so clamd starts at boot time. Otherwise, you need to create and enable your own startup script.

Configuring Amavisd-new

As with clamd, we need to edit only a single file, `/etc/amavisd.conf`, in order to configure amavisd. However, we've got a little more work to do here. Listing 2 shows the most important settings in my `/etc/amavisd.conf` file.

Listing 2. Important Settings in `/etc/amavisd.conf`

```
$daemon_user = 'amavis';
$daemon_group = 'clamav';
$mydomain = 'wiremonkeys.org';
$MYHOME = '/var/amavis';
$QUARANTINEDIR = '/var/virusmails';
$db_home = "$MYHOME/db";
$helpers_home = "$MYHOME/var";
$pid_file = "$MYHOME/var/amavisd.pid";
$lock_file = "$MYHOME/var/amavisd.lock";
$log_level = 2;
$virus_admin = "mick@$mydomain";
$mailfrom_notify_admin = "antivirus@$mydomain";
$mailfrom_notify_spamadmin = "antivirus@$mydomain";

### http://www.clamav.net/
['ClamAV-clamd',
 \&ask_daemon, ["CONTSCAN {}\n", "/usr/share/clamav/clamd.sock"],
 qr/\bOK$/, qr/\bFOUND$/,
 qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
```

The first two settings in Listing 2, `$daemon_user` and `$daemon_group`, specify the user and group amavisd should run as, `amavis` and `clamav`, respectively. As I mentioned previously, I like to use a common group for all of amavisd's and clamd's files, so it makes sense to have amavisd run as that group too. The `/etc/passwd` entry for my amavis account looks like this:

```
amavis:x:53:52:Amavisd-new Daemon:/var/amavis:/bin/false
```

`$mydomain` specifies your organization's name domain. `$MYHOME`, which should be set to the same path as your amavis account's home directory, specifies the root path to Amavisd-new's files, customarily `/var/amavis`. This directory should be owned and writable only by root. `$QUARANTINEDIR` is the path to the directory in which you want amavisd to dump quarantined e-mail messages. This directory should be owned by your amavis user account and writable only by it.

`$db_home`, which you may need to uncomment, specifies where amavisd should keep its databases, such as include cached scan results, among other things. `$helpers_home` specifies the directory in which you want amavisd to keep its SpamAssassin settings and other miscellaneous things. `$helpers_home` may be commented out by default. The directories specified by `$db_home` and `$helpers_home` should be owned and writeable only by your amavis user account.

`$pid_file` and `$lock_file`, also possibly commented out, can be used to specify where amavisd writes its pidfile and lockfile, respectively.

`$log_level` specifies how verbose amavisd's log messages should be, expressed as a number between 0 and 5, with 5 being the most verbose. The default is 0, but I find 2 to be a much more useful setting that nonetheless doesn't clutter my logfile. By default, amavisd sends its log messages to the local syslog system under the mail facility. In other words, your amavisd log messages should turn up in the same place as where your Postfix messages are written.

The next four settings concern e-mail originated by amavisd when a virus or spam message is detected. `$virus_admin` is the e-mail address to which you'd like virus notifications sent. This must be a valid e-mail address; update your local aliases file if the value you set here isn't yet. In practice, this probably should be the e-mail address of you, the system administrator.

It's also possible to configure amavisd to send notifications to each message's intended recipient or to its sender, but most people find this annoying, especially because the from addresses of spam and virus e-mails nearly always are forged. Don't do that.

`$mailfrom_notify_admin` and `$mailfrom_notify_spamadmin` specify the from address to use when amavisd sends virus or spam notifications, respectively.

Finally, we come to amavisd.conf's real magic: the antivirus scanner definition for ClamAV. In my default `/etc/amavisd.conf` file, this entire section was commented out, so I first had to delete the initial `#` on each line. You may or may not need to do this yourself.

You do, however, need to check the clamd socket path in this section. In Listing 2, I've changed mine from the default of `/var/run/clamav/clamd` to `/usr/share/clamav/clamd.sock`, the same path I defined in `/etc/clamav.conf`.

Once you've edited `/etc/amavisd.conf` and set the permissions on amavisd's directories accordingly, you can start the daemon by entering the command `amavisd` without arguments. As with clamd, you need to enable and maybe even first create a startup script for amavisd so it starts at boot time. I advise setting this up so that clamd starts first. That way, by the time amavisd starts, clamd's socket already is in place.

Also, as with clamd, you should start amavisd as root. It demotes itself to the user and group you specified in amavisd.conf.

Postfix Configuration

Our ClamAV and Amavisd-new daemons are configured and running. Only a couple more tasks remain, configuring Postfix for content filtering and updating ClamAV's virus databases.

Important note: the following assumes that Postfix already is configured for and successfully performing its normal receiving/forwarding duties.

First, open `/etc/postfix/master.cf` with your text editor of choice, and add the lines in Listing 3 to the bottom of the file, if they aren't there already.

Listing 3. Lines to Add to `/etc/postfix/master.cf`

```
smtp-amavis unix - - y - 2 smtp
-o smtp_data_done_timeout=1200
-o disable_dns_lookups=yes

127.0.0.1:10025 inet n - n - - smtpd
-o content_filter=
-o local_recipient_maps=
-o smtpd_client_restrictions=
-o smtpd_helo_restrictions=
-o smtpd_sender_restrictions=
-o smtpd_recipient_restrictions=permit_mynetworks,
  reject_unauth_destination
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
-o smtpd_error_sleep_time=0
```

```
-o smtpd_soft_error_limit=1001
-o smtpd_hard_error_limit=1000
```

The smtp-amavis section defines Postfix's outbound communications, using the SMTP protocol, with amavisd. It corresponds to the following line you should add or edit in `/etc/postfix/main.cf`:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

This line tells Postfix to send all incoming e-mail to 127.0.0.1, the local system, on TCP port 10024, amavisd's default SMTP listening port, by using the smtp-amavis interface we defined in `master.cf`. You can change amavisd's listening port by editing the `$inet_socket_port` parameter in `/etc/amavisd.conf`.

The second section in Listing 3 defines the inbound interface on which Postfix should accept messages returned by amavisd. In other words, Postfix listens on the local loopback IP 127.0.0.1 on TCP port 10025, which are the address and port to which amavisd sends notifications and forwarded messages by default. You can change amavisd's notification and forwarding address and ports by editing the parameters `$notify_method` and `$forward_method` parameters, respectively, in `/etc/amavisd.conf`. After editing `master.cf` and `main.cf`, you need to restart or reload Postfix.

Testing the System

Before we go any further, let's test the system. The simplest way to test is to send yourself an e-mail message containing the following string, which is not a real virus but a test string called the Eicar Test Signature:

```
X50!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

If everything is working, amavisd sent an e-mail to the account you specified in `amavisd.conf`'s `$virus_admin` parameter, and the message should be quarantined in the directory specified in `amavisd.conf`'s `$QUARANTINEDIR` parameter.

I highly recommend tailing your mail log while performing this test. Type `tail -f /var/log/mail`, and Postfix and amavisd will log their actions there. In my own experience, this is the fastest way to identify problems, especially if you increased amavisd's log-verbosity as described earlier.

Also, be sure to do at least one test with clean e-mail to ensure you haven't impaired Postfix's ability to receive and deliver unfiltered mail.

Updating ClamAV's Databases

There's only one thing left to do, but it's important: update ClamAV's virus-signature databases and create a cron job to do so automatically every day. ClamAV includes a utility called `freshclam` for this purpose.

Because using `freshclam` is the simplest task in this entire undertaking, and because I'm basically out of space for now, I leave it to you to explore the `freshclam(1)` and `freshclam.conf(5)` man pages. Suffice it to say that in normal practice you use the command form `freshclam -l /path/to/logfile`, where `/path/to/logfile` specifies the file to which you want `freshclam` to write its logs.

It's recommended that you run `freshclam` every couple of hours. The easiest way to do this is by running `freshclam` in `dæmon` mode via the `-d` and `-c` startup options. See the `freshclam(1)` man page for more information.

Conclusion

With that, you now should have a ClamAV-enabled SMTP gateway or at least be started down the road towards one. If you're having problems, the on-line Resources includes additional Postfix plus Amavisd-new tutorials. Good luck!

Resources for this article: www.linuxjournal.com/article/7811.

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](#)

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

Advanced search

Linux for Suits

Unusual Suspects

Doc Searls

Issue #128, December 2004

Like other rapidly growing sites, Technorati is extending the LAMP suite with Memcached, the Spread Toolkit and other independent spirits.

Building infrastructure for a search engine like Technorati's, which scours and archives the "live" Web of RSS and similar feeds, is like building a series of barns, each after the last one burns down. When recounting the brief history of his company, David Sifry usually says, "Well, after the first infrastructure fell down...and after the second infrastructure fell down...and after the third infrastructure fell down...." No wonder he also says, "Scaling is everything." Instructive failure is a less-glamorous description of trailblazing.

Technorati's trail started in David's basement, on a server that Penguin Computing loaned to David and me for a feature on blogs for the March 2003 issue of *Linux Journal*. See "Building with Blogs" and "The Technorati Story" in the on-line Resources. The service instantly was popular and has continued to grow steadily. By the time you read this, Technorati will be watching nearly four-million syndicated sources, mostly Weblogs, and well over a half-billion links. (Disclaimer: I'm on Technorati's advisory board.) Think of Technorati as a search engine for stuff that's too new for Google—plus a platform for free and paid services based on the company's growing archives and countless potential forms of derived data.

One can look at Technorati as another fast-scaling LAMP hack. But when I talk to Technorati's techies, it turns out some of the most interesting applications are from open-source suspects outside the familiar LAMP list of Linux, Apache, MySQL, PHP, Perl, Python and PostgreSQL. I recently asked the company's new VP of Engineering, Adam Hertz, who came over from Ofoto in July 2004, to tell me more about one or two of those useful but low-box-office tools. He

immediately pointed to Memcached and handed the explaining over to Ian Kallen, senior architect on his staff.

“A lot of what we're doing has never been done before”, Ian told me. “But a lot of the scaling problems we hit along the way *have* been experienced before. And the result is open-source freeware, often very robust, that eases exactly the pain we're experiencing.” One such result is Memcached, a distributed caching system developed by Danga Interactive, the folks behind LiveJournal, which accounts for a very large percentage of the blogs Technorati follows. Ian explains:

Lookup queries in a large data repository can be expensive. If lookups are repeated at a rate near or less than the rate of change, it makes sense to spare the application repeated round trips to fetch the data. An obvious solution is to have the application process keep lookup results in a cache. The downside is that caching is typically resource consumptive; in a multiprocess application such as a Web service, having each process (or thread) keep its own cache reduces the cache efficiency and bloats the application resource footprint. The solution to this is a shared cache.

High application availability requirements are typically met with server redundancy. Now if each server instance has a cache of its own, the cache efficiency reduction becomes as ridiculous as having individual processes share a cache on a single host. The solution there is a distributed cache and that's where Memcached comes in.

Listening on a network socket and utilizing very simple parameters to instrument memory allocation, Memcached maintains an in-memory dictionary of keys that is dynamically populated with values. Technorati uses search parameters as keys and search results as values. The payback comes when search results that are duplicative of previously run queries return; they come back at least an order of magnitude faster.

In addition to its speed and simplicity, Memcached's other principal strength is its flexibility. It can store objects in language-native serialization formats such as Perl's Storable or Java's Serializable. Technorati uses PHP's native serialization to store cache results. However, Memcached can store any bytes that can be used interoperably; one cache client implemented in Java can access and read a cached XML document stored by another client written in Perl, Python or PHP. This can be a huge win for heterogeneous development environments.

Application stabilization and performance optimization is a critical concern for burgeoning data repositories such as Technorati's. Caching isn't the only thing in the software toolbox. Application architectures can still benefit or suffer from the quality of other design decisions. However, the integration challenge posed by Memcached is so low as to make it a primary consideration for data retrieval acceleration.

When I asked Adam again about other less-known tools that might be in his box, he said the company also was about to use the Spread Toolkit—a language-independent messaging system that allows updates, events and information to flow through distributed systems. He explains:

When a blog pings us, think of that as an event. Our spider responds and goes and gets the new content. It then puts the update on the message bus, so any application can see the message as it goes by. Each subscriber—a Technorati application or service—gets a chance to see every message that goes by. It can either pick it up or pass on it. An application can say, “hey, am I interested in this update?” This way, we reduce the internal query load on the database, and we also make the applications more real-time. This is called multicast. It's a service similar to what TIBCO has been for financial systems.

For example, a blog post about a political book might be interesting to Book Talk, News Talk and Politics Today. Rather than having all three applications querying the database to find relevant updates in the last five minutes, that blog post would travel across the system and be picked up by all three applications.

This not only speeds up performance for users and applications, but opens lots of new service opportunities for users, outside applications and services using the Technorati API.

Jonathan Stanton, one of Spread's creators at Johns Hopkins University, added this:

Spread (also) provides fault-tolerant messaging. Meaning if some machines that are part of a Spread group crash or are partitioned from the others, Spread guarantees that all the machines that are still connected will see the same set of messages. This strong semantic group messaging can be used to build replicated databases (one open-source project building a replicated database using Spread is Postgres-R) and guarantee consistency in clustered caching systems (like Technorati's).

Five interesting things here. First, note where the Technorati *doesn't* look for answers. The old industrial model assumes that you obtain the expertise you need internally (from a responsible position on the company org chart), or you buy products and expertise from outside vendors and consultants. Here, we have a CIO and a lead engineer looking outside the company for infrastructural building materials, plus experts and expertise that are both *in* the marketplace yet *not* for sale. In other words, tools like Memcached and Spread are part of a larger conversation, and a larger set of relationships, than a corporate customer like Technorati would get from a commercial vendor. This isn't a knock on vendors, but it reminds us how markets defined in terms of vendor relationships, and of sales volumes in product categories, fail to include a large and growing part of the market ecosystem.

Second, the advantages of using open-source tools, and of participating in development projects like Memcached and Spread, are likely to be lost on traditional IT shops that discourage trailblazing DIY work. Adam Hertz explains:

Two themes: BigCoIT is all about standardization and isolationism.

Standardization, so the story goes, reduces risks and costs. It certainly reduces complexity, but it can take a huge toll on flexibility and responsiveness. Standardization often involves using one multipurpose tool or platform to accomplish lots of different purposes. This often involves customization, which is done by in-house experts or professional services firms. Great examples are Siebel, Lotus Notes and so on.

DIY shops tend to be cynical, or even downright frightened, of systems like that, because they're so inflexible and unhackable.

Another form of standardization is what people are allowed to have on their desktop PCs. In a lot of big shops, everyone has the same disk image, with all applications pre-installed. There's a huge suspicion of anything that comes from the outside world, especially open source. It's regarded as flaky, virus-laden, unscalable and so on. This produces isolationism, which means that there are major barriers merely to try something.

In more open environments, there's a permeable membrane between the corporate IT environment and the Net. People tend to get new tools from the Net, usually open source, and just give 'em a spin. Culturally, this keeps the organization open to innovation and new approaches. It builds bonds between the employees and the development community at large.

Standardized, isolationist shops miss out on all of this. They maintain control, but they inevitably fall behind.

Third, Technorati isn't only solving problems, although problem solving does soak up a lot of IT cycles. It's also looking to tools like Memcached and Spread to open new business and other opportunities. When I asked Adam about using Spread to expand Technorati's Web service offerings, he said the opportunities, already wide open, only get wider:

It would also be very useful to implement customized watchlists, for example. Suppose you had some weird-assed query...or more accurately, filter. Like, very specific sorts of posts you want to find out about—posts that mention you, *Linux Journal* and open source, for example. We could set that up as a subscriber to updates. It could watch all the updates come by, discard the 99.99% of updates that aren't relevant, and when it gets a relevant one, it could send you mail or something. Contrast that with our current watchlist implementation that queries the database. Then and imagine if—or when!—we have 100,000 watchlists.

Fourth, Adam is quick to defer to his engineers' expertise, among which is finding useful and free open-source tools in the marketplace. The sounds Adam makes when he talks about his work remind me of those Linus Torvalds makes when he talks about kernel maintainers. Linus said:

It is very hard to find people who don't flame and are calm and rational—and have good taste. I mean it's like...give me one honest man. It doesn't happen...too much. And at the same time, when it happens, it matters a lot. Just a few of these people make a huge difference.

Fifth, it becomes clear that trailblazing work by tasteful flameproof engineers is what gives the world the infrastructure it needs to support real business, in addition to maintaining the mundane internal operations of enterprises.

Just in the last few days, while I was writing this column, two other open-source tools came to my attention. They're interesting because they represent two extremes in the marketplace: one external, one internal.

The external one is the Gallery Project, which appears to be establishing itself rapidly as the Apache of photo gallery software. Within a week, Gallery went from something I barely knew to something it seemed everybody was telling me about. It's an extraordinarily rich and deep system that surely will be far more rich and deep by the time you read this. It shows that Apple, Adobe and Microsoft aren't the only ones who care to provide users and photographers

with useful tools—and that the tools that matter most aren't those that live on clients or vendor hosting services, but rather on photographers' own servers, or those of their friends, relatives and other parties, including businesses. Gallery is pure DIY infrastructure and another dramatic example of how the Net and open source allow the demand side to supply itself.

I also think it's significant that Galley scratches an itch that isn't only technical. Although it's a bit of a hack to install, it's what marketers call a consumer-facing service. It will be interesting to watch the market effects over the coming year or two.

The internal tool is Yum (Yellowdog Updater Modified), an automatic updater and package installer/remover for RPM-based distributions, developed by the mathematics department at Duke University—specifically by Seth Vidal, the Senior Systems Programmer there.

Yum is public code, with public source repositories and a GPL license. Yet it was developed by Duke for its own internal requirements. Richard Hain, chair of the Math department, recently told me more than 100 Linux machines run in that department alone, with a serious commitment to running Linux on the desktop, as well as on larger machines for serious number crunching. Most office staff are running Linux desktops. Microsoft Windows users are running on Win4Lin. Yum is a necessity in their environment, Hain explains. Frequent hardware updates alone require a “departmental distro that runs Yum every day and maintains packages”.

So, it seems to me, the unusual suspects that matter most in open-source markets aren't only the tools, but the independent spirits that create and use them.

Resources for this article: www.linuxjournal.com/article/7806.

Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal*. His monthly column is Linux for Suits and his biweekly newsletter is SuitWatch. He also presides over Doc Searls' IT Garage (garage.docsearls.com), a sister site to *Linux Journal* on the Web.

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

[Advanced search](#)

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

[Advanced search](#)

EOF

gnuLinEx: Foundation for an Information Society

Dario Rapisardi

Issue #128, December 2004

Thanks to an operating system that empowers students to develop skills, and an ambitious public access plan, the Internet isn't leaving the poorest region in Spain behind.

People who visit Extremadura are nicely surprised at the rich history of the place, the kindness of the people and the good cuisine. Now we can add a new characteristic to the qualities of the region: it has the largest Free Software implementation in the world.

Extremadura is one of 17 autonomous communities in Spain. Each one of these communities is made of one or more provinces and has considerable independence. They are entitled to rule and decide independently from the central government on several home affairs, including their educational systems.

Extremadura also is the poorest region in Spain and the fifth poorest region in the European Union. In its recent history, we have seen how diverse factors have increased the economic and social breach between the region and the rest of Europe. In 1998 an alert arose—the Internet was going to increase the gap.

For that reason, a unique project was started. A regional network called Intranet Extremadura was created, and it currently connects more than 1,400 dispersed points including regional administrative centers, schools, hospitals and centers for public access.

But that is not everything. Since 2002, Extremadura has the largest number of computers per student in Europe, with an average of one computer for every

two students. The number goes up to 68,000 computers serving more than 120,000 students in high schools.

All these computers make up a network running the favorite operating system of the readers of this magazine, GNU/Linux. The localized version is called gnuLinEx, formerly LinEx.

From its beginning, the distribution has been based on Debian GNU/Linux, with the effort focused on the adaptation of the operating system, translation of educational applications and the simplification of the installation process.

Last August, the 2004 version of LinEx appeared. The Debian port of Anaconda made by Progeny is used for the installation. Any GNU/Linux user that previously had installed Red Hat or Fedora before already should be used to it.

We offer either a Gnome 2.6 desktop or XFCE4 for low-end computers. The kernel version is 2.6.7 with support for diverse devices found in Spain and patched with supermount to allow easy access to removable devices from the desktop.

From the educational applications available in gnuLinEx, the most impressive one is Squeak. Squeak is a powerful multimedia and educational environment—an implementation of Smalltalk-80, whose image for the Spanish-speaking world is contributed to actively by the government of Extremadura itself.

The new features of gnuLinEx are not only for the end user. This distribution is one of the first to use component technology for the building of the distribution. This technology, part of the Componentized Linux (CL) Project, allows us to think of a GNU/Linux distribution as a set of interchangeable parts or, in terms of CL, components.

This way, a distro builders can add support for MySQL simply by adding the respective component, or change the GNOME 2.6 component with a KDE 3.2 component if they prefer.

The base component of CL, also included in gnuLinEx, is called LSB 2.0. It contains all the necessary software to comply with specification 2.0 of the Linux Standard Base (LSB).

But the component model did not only help in the building of gnuLinEx; it keeps helping in the building of derived works as well, such as LinEx Company and LinEx Gamer. The first one is oriented to the business world, and the second one is for all of us who enjoy using the computer for more than work.

Guadalinex is the localized version of Debian GNU/Linux used by the autonomous community of Andalusia, Spain. The people of Andalusia and Extremadura reached an agreement to unite their efforts in the creation of their respective operating systems for the 2005 version, with a considerable common base. Dividing efforts in the maintenance of their own components, both teams will be able to concentrate on other subjects such as accessibility, usability and hardware support.

If you are a Spanish-speaking GNU/Linux enthusiast and want a complete Debian system with graphical installation, or if you are interested in the use of computers in education, you should take a look at gnuLinEx 2004.

Mérida is the capital of Extremadura. Two thousand years ago, it was the capital of the Roman province of Lusitania and one of the most important cities in the Roman empire in Emperor Augustus' time. Currently, it is starting to be considered the capital of the Free Software empire, with the largest implementation worldwide. You are invited to join.

Resources for this article: www.linuxjournal.com/article/7819.

Dario Rapisardi left Argentina following the gnuLinEx dream and the good ham to settle down in Spain. He currently is a gnuLinEx developer and can be contacted at dario@rapisardi.org.

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

[Advanced search](#)

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

[Advanced search](#)

Monarch ULB 64 2005 Custom Workstation

Chris DiBona

Issue #128, December 2004

Product Information.

- Vendor: Monarch
- URL: www.monarchcomputer.com
- Price: \$6,087 US

The Good.

- Quiet as a tiptoeing mouse.
- Super fast.
- Reasonably priced.
- Very expandable.

The Bad.

- Linux load not terrific.
- Sound needed configuration on review machine.

The Review Machine.

- Two Series 250 processors.
- 2GB memory.
- 3ware serial ATA (SATA) RAID card.
- Four 74GB SATA drives.
- NVIDIA Quadro 3000 video card.
- Red Hat Workstation 3.
- Plextor px-712a bl DVD Reader (16×), Burner (12× and 8×).
- Asus CD-ROM Reader [52×], Burner [52× (r) 32×(rw)].

- 7-in-1 floppy/memory card.

In the October 2004 issue, I reviewed an IBM machine not dissimilar to this one. The IBM I reviewed had some very real problems and I noted them in the review, but had I received this machine first, I can only imagine how much harsher I would have been to the IBM. In a way, I'm glad I was able to review the IBM first, as it wasn't all that bad a machine. But, when considering the purchase of a high-end workstation, you need to comparison shop among large vendors, such as IBM and HP, and smaller vendors, such as this system's builder, Monarch.

The Monarch machine arrived at my house packed snugly in a huge, imposing box that had been nestled carefully into a larger box packed with more Styrofoam. The machine, constructed within a black Lian Li case, could be described only as beautiful. If I had any machines that were stuffed into typical beige boxes, the beauty of this case would make me want to take them outside and smash them with a hammer.

But for me, the external beauty of a machine is not even on my list of top ten features of a machine. A computer is meant to be used, and the case is often meant to exist, gathering dust, under my desk, where it's accessed rarely to pop a CD in the drive or to power off.

As is my habit, the first thing I did when it arrived, having removed it from its cardboard, plastic and Styrofoam womb, was to pop open the case. This is something the Lian Li engineers really get. One screw and the interior of the machine is wide open for your inspection—and what an interior! Although other vendors have caught on to the importance of routing cables properly, Monarch really does a nice job of this, and it's always nice to see it.

The inside of this machine was as beautiful as the outside, and the hardware selected for this machine was top-notch. For storage, four 74GB SATA drives were situated neatly in a drive bay along the bottom of the case, with two optical drives and a 7-in-1 floppy/memory bay in the user-accessible bays in the front. The case allows for an additional two drives in the lower drive bay (for a total of six) and an additional three or four (depending on the model or the determination of the installer) devices or hard drives in the user-accessible bays along the front of the case. As you can see, this machine has ample room for expansion from a storage perspective. The SATA cables are tied off to prevent tangling.

To support all those drives, the machine comes with a 3ware card as well as the motherboard-provided IDE controllers. The Opterons are each cooled with a ThermalTake CPU cooler, and there are two banks of four memory slots each.

Topping all of this off, the machine ships with the top-notch NVIDIA Quadro 3000 video card.

Remembering my last review, I think I compared the IBM A Pro to a jet taking off, so if the Monarch was anything but a dull roar, I would have been happy about it. So, I plugged it in and turned it on to find out.

At first, I wasn't sure I had turned it on. I killed my music and then I heard it. It was slightly louder than the small Shuttle-based workstation that I keep tucked under my desk. I communicated this to my editors and they shipped me the sound pressure level (SPL) meter that *LJ* keeps around for such things. My Intel 2.4GHz desktop measured around 39 decibels, and this monstrously powerful machine came in at slightly more than this, 41 decibels. During some of the more powerful apps I threw at it, it reached 44 decibels. To make a comparison, when I was speaking with my three-year-old, she and I came in at around 48 decibels. But, enough about the mechanical and the construction—how does it run?

The Software

Keeping in mind I have the prejudice that most users of this kind of machine instantly blow away any factory load and put in their own, I was pretty happy with the software load—your standard Red Hat 64-bit workstation load helpfully set to a default 800×600 resolution for initial configuration. I was able to bring the resolution up and put X in a high and deep resolution. The system ships with the NVIDIA driver installed, which is a good thing if you want to take full advantage of the serious video card that comes with it.

As you could predict, given the video card the machine shipped with, video and OpenGL performance was very smooth, with the pre-installed applications running very well. But, when I went to build some test applications, I was unable to make a compile complete. The load was not set up for compiling, which is a minor problem, but as a reviewer, it was one I wish I didn't have to worry about. I had no problem getting binaries of some sample applications running on the machine and experienced a smooth ride on *Tux Racer* and others.

That said, I really like to be able to compile apps, as many 32-bit builds don't run as well as they could otherwise. Considering that 64-bit RPMs still are unavailable for many applications, this should be a standard feature. That said, the machine ships with installation media, so properly loading the features I wanted wasn't a showstopper.

Like the last machine, the sound card was not configured, but a quick run of `redhat - config - soundcard` fixed that quickly. Unlike the last machine I tested, the sound was crystal clear, with no pops, hums or clicks.

The fantastic, sexy, amazing NVIDIA Quadro 3000 FX video card was equipped with dual-DVI outputs, and Monarch thoughtfully included DVI adapters. I tested the dual-monitor modes to good result.

The machine also shipped containing a handy portfolio with all the manuals, warranty information, passwords and CD-ROMs that came with the individual components, which I also really liked, as it is often handy to have such things around for unforeseen uses of the machine.

In short, if you are looking for a graphics workstation, you could do a lot worse than the Monarch. Damn, this thing is quiet.

Chris DiBona is the Open Source Programs Manager for Mountain View, California-based Google, Inc. These writings are the author's opinion and don't necessarily reflect those of his employer. Before joining Google, Chris DiBona was an editor/author for the popular on-line Web site slashdot.org, and he is an internationally known advocate of open-source software and related methodologies. He co-edited the award-winning essay compilation *Open Sources* and can be reached via dibona.com.

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

[Advanced search](#)

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

[Advanced search](#)

From the Editor: December 2004

Great Entertainment Software

Don Marti

Issue #128, December 2004

Get started with video production, but keep your eyes open for sticky situations.

We normally don't bother with those "educational use only" disclaimers that other magazines slap on every article that might possibly be used for harm. Of course it's for educational use, and of course you're responsible for your own actions. But this issue, as we take on the tricky topic of entertainment, consider this an extended disclaimer.

Olexiy Tykhomyrov and Denis Tonkonog bring you an intro to Kino, a versatile editing package that supports plugins and uses XML to store its edit decision lists—the perfect tool for dealing with the high-quality footage from a digital camcorder (page 54). If you don't want to splash out for a camcorder, Marcel Gagné presents a smaller, cheaper, Webcam-based studio on page 30.

But, we have to admit that this issue is still missing a piece of the puzzle. Want to put the MPEG-4 videos you produce on a Web site that also has advertising? You'll need a lawyer and a contract with the ominous MPEG Licensing Authority. Where would the Web be if there were an "HTML Licensing Authority"?

There is hope. Check out theora.org for info on the patent-free Theora video format, which you can use as freely as HTML. We'll be bringing you more how-tos on staying patent-free; in the meantime, use MPEG for research only.

Although good news on software patents is still scarce, the software community, on both the free and proprietary sides, is aware of the danger and moving to contain patent damage. The Internet Engineering Task Force shut down its "Mail Transfer Agent Authentication in DNS" working group because of a threatened patent trap from one vendor. In the long run, that's the right

decision. Meanwhile, the Sender Policy Framework (SPF) Project, led by *Linux Journal* author Meng Weng Wong, is still going strong—and patent-free. Popular Linux distributions are also scrupulous about leaving off patent-encumbered software.

Readers were enthusiastic about Meng's spam-fighting articles, so we're bringing you more. Clam AntiVirus came out of the blue to win an Editors' Choice Award for best security tool this year. With non-Linux systems on the company network, your Linux mail server needs to protect them from viruses too. Mick Bauer gives you a detailed look at Clam on page 36, and Robert LeBlanc shows how to integrate mail filtering into an easy-to-manage system that lets users fish their own false positives out of quarantine on page 84.

The software patent mess won't be over until governments realize that they're a bad bargain—that the transaction costs and free speech risks outweigh whatever benefits come from R&D incentives. While you keep your eyes open and your “letter to politicians” template handy, and watch swpat.ffii.org for things you can do for patent reform, don't forget why you love digital freedom, and do some fun and useful things with the great software in this issue.

Don Marti is editor in chief of *Linux Journal*.

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

[Advanced search](#)

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

[Advanced search](#)

Letters to the Editor

Throne Reading

This is in response to the “Hey! You Kids! Get Out!” letter in the September 2004 issue. Geeks have kids, and we are as proud of them as any snippet of code. My kids *love* Linux from *Tux Racer* to GCompris to screensavers. They have their own Linux box in the works and have never been exposed to anything else. Heck—we even use Linux during potty training! Here is a pic to prove kids and Linux match up quite nicely, and that *LJ* is a *great* potty-training tool.



—

Mark

More Ham Please

Just a quick note to congratulate you for your *LJ* September 2004 issue. Yesterday, I went to my favorite newsstand and was immediately drawn by the great article written about PSK by Dr Volker Schroer. This article was the main reason I decided to buy the magazine. It is an excellent article and has inspired me to go back and experiment with PSK under Linux. I'm an amateur radio operator and was a former subscriber to *LJ*. I would love to see more Linux Ham radio articles!

Vy 73

—

Luis Delgadillo
XE2AC - W2GO

Achtung! SuSE Cat!

Serena is guarding my Linux distro archive.



—

Leigh

Another Penguin Beer

I was recently in Brazil, and thought you might be interested to see what beer they drink in the land of Conectiva.



—

Robert Rust

Tracking Down a Network Problem

I just purchased a Toshiba M30 laptop and installed Debian on it. It took three days to get it all up and running, but everything works perfectly, excluding the Winmodem. Shortly after everything was running smoothly, I ran into problems. Right after I had upgraded my Wi-Fi drivers, KDE applications started locking up randomly. They never crashed, but they would freeze for up to five minutes before performing a function. KDE would take ten minutes to load, but for some reason, this only would happen after the machine was running for a while. Naturally, I rolled back the drivers to no avail, then I assumed it was temperature, but the rest of the system seemed very stable and the machine wasn't that warm.

I really didn't know what to do but I recalled the "Ten Handy Commands Every Linux Developer Should Know" article from the September 2004 issue of *LJ* and decided to `apt-get install strace` and try to figure out what was causing my problems. I couldn't figure out the problem in KDE's startup sequence, but Konqueror gave away the problem when I saw it hanging on an attempt to connect to 127.0.0.1. I took a peek at my `/etc/network/interfaces` file to discover that I, or some script, inadvertently had removed the `iface lo inet` Loopback entry that provides the network loopback device. I added it back in and brought up the interface. Everything is back to normal. Thanks *LJ*.

—

Pete Osborne

Support for Laptop Hardware

I just finished reading Lincoln Durey's article "Dear Laptop Vendor" [*LJ*, October 2004], and I'm forced to ask "How does selling me (or Lincoln) a laptop without an OS on it help Linux run on it better?" Don't get me wrong, I'm one frugal guy, so I'll gladly take the \$100 discount, but how does that help (except my wallet)? Wouldn't it be better to ask for Linux drivers for their hardware or documented specs so that open-source developers could write drivers for the hardware? Perhaps I don't understand why Linux can't do all those laptop things, but isn't that how Apple's OS X can be UNIX and still be a fully functional laptop? Apple owns the hardware specs, and they write good drivers right? If not, how do they do it? Their laptops suspend, hibernate, wakeup, do back flips and make me breakfast.

My letter would be to the hardware vendor that's the closest to going out of business (more desperate and therefore hopefully more willing to take a chance), and it would say, "Want to boost sales? Write Linux drivers for your laptop so that all the bells and whistles of your platform will work. Oh, and if it's not too much trouble, dump the pre-installed OS and reduce the price accordingly!"

—

Anonymous

Where drivers are concerned, the biggest thing a laptop vendor can do is to require GPL drivers or unrestricted documentation from the chipset suppliers. Watch for a review of HP's first Linux laptop next issue. —Ed.

Help! A Penguin Ate My Quarters!

I called this photo "Subliminal invasion". They don't know what it is, but now, they know that it exists.



—

Francisco Ruiz

That's the new *Tux Racer* arcade game from roxorgames.com. Why not bug your local arcade manager to install one? —Ed.

Hurricane-Proof?

My house got demolished by hurricane Frances, but my Linux box booted and ran fine, and my *Linux Journal* arrived two days later. Why wasn't the car damaged? I believe the Tux logo chases away bad mojo.



Tom Maier

Which Desktop?

My 4.5-month-old son, Zach, and I arguing over GNOME and KDE for his first desktop.



Brian Cashman

Photo of the Month: a Ten-Year Collection

My fidelity to *Linux Journal* lasts nine years, and next month I'll start my tenth uninterrupted yearly subscription! From time to time, I revolve the magazines looking for that article I need. My wife captured that moment. The drink is a mate, a kind of wild tea of Argentina, Paraguay and southern Brazil. Hope to send the same picture when I am a 20-year *LJ* subscriber. Keep doing everything the same way. Good work—especially for us, south of the Tropic of Cancer, where Open Source is the right (only?) choice.



—

Guillermo Giménez de Castro

Photo of the month gets you a one-year extension to your subscription. Photos to info@linuxjournal.com. —Ed.

Why DIY?

I've just read Phil Hollenback's article on building a Linux-based WAN router. I was wondering if there was a reason why he didn't use ImageStream's Rebel Router, which was reviewed by *Linux Journal* several years ago [August 2002]. Keep up the good work.

—

N. Huggins

Phil's reply: that's a good question, which I probably should have addressed more directly in the article. We did examine the ImageStream routers. However, we didn't go with them for a couple of reasons. 1) Most important, their drivers are proprietary and closed. Plus, they make extensive modifications to the Linux kernel to make their drivers work. We didn't want to

use proprietary software. 2) I'm a little hazy on this because it has been a few months; however, I believe the ImageStream hardware that will support a T3 plus four T1s all in the same router gets to be quite expensive—their basic Rebel Router can't handle that much bandwidth. You have to move up to their carrier-grade solution.

The bottom line is we customize the heck out of our infrastructure, and we wanted the software and hardware with the most flexibility. That's why we built our own. However, if you are looking for a basic router and don't have the admittedly esoteric requirements that we did, the ImageStream Rebel seems like a good choice. Thanks for reading the article!

Revealing Hidden Commands

Thanks for the article “Ten Handy Commands Every Linux Developer Should Know” [*LJ*, September 2004]. I'm not a developer, but a now-retired longtime UNIX administrator. I'm familiar with most of those commands but have been missing my favorite `ofiles`, and now I see `fuser` is a command with the same purpose. And, I was unfamiliar with `objdump`. Another program I used to use a lot—I've forgotten the name at the moment—took a dump of a running program without disturbing the program that was running. All of this points out that Linux has a lot of neat stuff that people may be unaware of. I know when I install a Linux distribution, I see stuff getting installed that I don't know what it is, and I have never used it.

—

jhhaynes

***LJ* for Kids?**

I am delighted how many people send their children's pictures to be published. Think about your readers' demography. Don't you think that it would be a great idea to have an issue dedicated to kids and young teenagers? Please, let me know if you read this and I will send you a picture of my one-year-old son in front of the computer.

—

Diego Betancor

We'll keep it in mind. You and your kids can have fun with Kino (page XX.)—Ed.

Maybe We Should Rename This Section...

Here is our two-month-old daughter Avery being very happy with Tux. I've been inching into using Linux at work and have started documenting our progress.

Our most popular page (by visits) is www.flmnh.ufl.edu/linux/fc1_intel_video_fix.htm.



—

Dan Stoner

...to **“Babies and Penguins”**

Here is a picture of my little girl, Madeleine. As you can see, she is well on her way to creating open source software. Well after lunch that is.



—

Christopher Graham

Market Opportunity?

Let's face it, Linux is not a user-friendly, ready-for-the-masses system. I'm thoroughly convinced that Linux must get on the desktop for the masses. The longer it isn't, the more time Microsoft has to wheedle into the server market and hijack the Internet. Today, Linux hasn't made a strong enough push on the desktop. Microsoft is so confident on the desktop that their cheapest offering now retails for \$200! A lot of boats are being missed by Linux these days. Much has been made of the network or appliance computer, a cheap, minimalist home machine that would do Web, e-mail, digital photography management and word processing (all 90% of home users really want/need from a computer). It's wide open for Linux, but Linux seems to not care.

—

Olwe

Why not start the next great home computer company yourself? Twenty percent of Linux is "U". —Ed.

Laptops, Security and Thin Clients

Lincoln Durey's "Dear Laptop Vendor" [*LJ*, October 2004] was not very useful. It would have been a far better use of space, and his expertise, to explain how his staff sorts out and solves the problems with getting Linux to work properly on the laptops his company sells, and what sorts of changes need to be made. Software is an add-on option with ChemBook laptops. Durey already has a solution to his problem if he wants it.

As a beginner/intermediate user I found Mick Bauer's clear and complete explanations enjoyable to read as well as helpful. I look forward to part two of Linux Filesystem Security [page XX]. I appreciate very much that Mick Bauer neither assumes I know something, nor talks down to me, nor dumbs down his article.

Please do a thorough article on how to create thin clients and serve them with a Linux server. Linux is inherently a server OS. This huge advantage over the popular proprietary OSes is not being exploited to its full advantage for desktop use. There is no reason for a home or small office to buy all new computers every couple of years. All that is necessary is to buy one good computer and use the old computers as thin clients. I've seen it done, and now I want to know how to do it.

—

Eric Skalwold

Watch for coverage of thin clients and other desktop management topics in our February 2005 issue. —Ed.

Mention the US Army Connection

I'm really astonished by the little article by Gary Glasscock on the game *America's Army for Linux*. I'm a peace activist here in Italy. Just to be clear, I've found no problem at all if Linux and free software are used for military purposes, and I've found *LJ* very equal on these subjects, writing about "Tux at war", but also writing about "maddog" Hall on Brazil or the digital divide and Africa. So, I'm astonished because a single thing is not stated in this article: that the site www.americasarmy.com is an initiative of the US Army, I think primarily for recruiting troops. I think this is a real mistake; please, preserve the *LJ* impartiality that I like so much. Many thanks.

—

Marco Gaiarin

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

[Advanced search](#)

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

UpFront

- [diff -u: What's New in Kernel Development](#)
- [LJ Index—December 2004](#)
- [On the Web](#)
- [They Said It](#)
- [The Battle for Wesnoth:](#)

diff -u: What's New in Kernel Development

Zack Brown

Issue #128, December 2004

After almost a year of rivalry, **pmdisk** and **swsusp** will be re-unified into **swsusp**, and **Pavel Machek** will continue to lead **swsusp** development, with help from **Patrick Mochel** who did the fork and other folks interested in the software suspend feature. Patrick made a nice apology to Pavel for forking the code, and the two have been working very nicely together since then, sharing patches and conversing on the technical issues. It's nice to see this dispute resolved, because it all emerged from a shared passion to make the code as good as it could be, to provide a solid software suspend feature and to merge the work into the official kernel with as little breakage as possible.

cryptoloop is very likely to be dropped from 2.6 unless **Andrew Morton** decides to wait to remove the code in 2.7; either way, it seems clear that **cryptoloop** is on the way out. The feature, intended to allow the user to mount encrypted filesystems over loopback, is apparently buggy and unmaintained, as well as having significant security problems. And as far as Andrew is concerned, it is this security question that dooms **cryptoloop**. Better to have no feature, he says, than a feature that falsely claims to provide real security. The argument, put forward by a number of developers, that such major changes as removing a feature should never occur in a stable series, is being heard, but it seems that this may only delay the inevitable. Unless someone steps up to maintain it and fix the security holes, **cryptoloop** probably will not last much longer.

The **Reiser4** filesystem, a significant departure from **Reiser3**, has been accepted into the -mm kernel series (Andrew Morton's personal tree) and appears to be on the fast-track for inclusion in an official release. The Reiser folks are ecstatic about this, as it tremendously widens the field of users and thus, of bug reports. Reiser4, with only the core functionality submitted to Andrew, claims to be the fastest filesystem in the Linux world. That claim may involve quite a moving target, but undoubtedly this new generation of filesystems is picking up speed in general, which makes a huge difference to anyone involved in disk-intensive operations.

Andrew Morton, as the 2.6 maintainer, has decided to question some recent traditions, including the validity of having a stable and unstable kernel series. Although his innovations may not be so extensive as to discard the entire idea, he has stated that he feels it is in the realm of the distributions to bring true stability to the kernel. The official kernel sources, he says, should focus on speed and features as well as stability, but not give stability the high pinnacle of importance that it has been given for the past several major releases. There are many ways to think about this, and it's important to remember that Andrew, Linus and the rest always are modifying their development model in response to their own ideas and the ideas of others. Perhaps the most optimistic way to look at Andrew's current idea is as a way to expand the field of kernel development to include the distributions as full-class members of the development process, with stability as one of their primary aims.

DevFS, having been at the center of one of the largest controversies in Linux history, seems to be edging closer and closer toward finally leaving the kernel. Andrew Morton has said that the 2.8 kernel certainly will not have DevFS, and that the feature could be removed as early as mid-2005. There always has been a push to keep DevFS in until its replacement, **udev**, could provide all the features DevFS provided. For all its faults, DevFS has proven to be a tough feature to replace. **Richard Gooch** put a tremendous amount of work into it, and in spite of all the criticism he got from **Alexander Viro** and others, his work still has not been replaced, almost two years after Richard abandoned the project and gave up on kernel development entirely.

LJ Index—December 2004

- 1. Number of digits of growth projected for Linux: 2
- 2. Percentage of UNIX users who have a desire to switch platforms: 4
- 3. Percentage of Microsoft Windows users who have a desire to switch platforms: 10
- 4. Millions of developers in North America working on open-source projects: 1.1

- 5. At least this many million developers in North America are working on 64-bit architecture: .5
- 6. Nearly this many million developers in North America are working on grid computing projects: .25
- 7. Nearly this many million developers in North America are working on clustered computing: .5
- 8. Percentage of computing developers in North America who are working on clustered computing: 17
- 9. Thousands of simultaneous voice calls voice calls per minute that can be handled by the Linux-powered Emergency Response System: 10
- 10. Thousands of simultaneous inbound hotline calls that can be handled by the ERS: 30
- 11. Thousands of simultaneous faxes that can be handled by the ERS: 5
- 12. Thousands of simultaneous text messages that can be handled by the ERS: 5
- 13. Percentage rate of the Internet's growth over 12 months ending June 2004: 26.1
- 14. Millions of new hostnames added during that period: 10.7
- 15. Millions of sites surveyed by Netcraft in June 2004: 51.635284
- 16. Millions of sites running Apache: 34.710235
- 17. Apache percentage of all sites: 67.22
- 18. Apache percentage increase over prior month: .17
- 19. Microsoft IIS percentage of all sites: 21.35
- 20. Microsoft IIS decline from prior month: -0.13

- 1-3: *LinuxWorld*, sourcing a Yankee Group report
- 4-8: Evans Data Corporation
- 9-12: Emergency Response Network
- 13-20: Netcraft

On the Web

If you want to read more from *Linux Journal* authors, add our Web site to your bookmarks or RSS reader and catch their Web articles and columns.

- If you're anxious to try Kino after reading "Making Movies with Kino", on page XX, but don't know how to get started with it, be sure to catch Olexiy Ye Tykhomyrov's follow-up article on the *LJ* Web site. In "Kino Tips: Installing from Scratch and Exporting MPEG Video" (www.linuxjournal.com/article/7615) Olexiy shows you how to get Kino on

your system and how to send your first film out to friends, family and Sundance.

- Following up on his idea that scouting and open-source philosophy share more than a little common ground, Marco Fioretti conducted a round-table e-mail interview with Richard Stallman and Ray Saunders, IT director of the World Scout Movement. Read “Bit Prepared II: Richard Stallman Meets the World Scout Bureau” (www.linuxjournal.com/article/7804) to learn what the connections are and how the free software community works together with this international organization.
- Ludovic Marcotte has written several articles for *Linux Journal* about GNUstep. His Web article “GDL2: the GNUstep Database Library” (www.linuxjournal.com/article/7101) describes GDL2, the “complete framework to develop database-oriented applications on GNUstep”, and walks you through coding an app that builds on either Linux or Apple Mac OS X.

They Said It

They are struggling with not so much open source, per se, but rather they are no longer the low price solution. In the past Microsoft was the low cost solution and Microsoft was then competing and attacking expensive proprietary systems from below. Now for the first time the tables are turned and it's Microsoft that's being attacked from below by a lower price solution.

—Brad Silverberg, former member of the Microsoft executive team (www.milestone-group.com/news/04_07/Silverberg.html)

There's no good DRM [digital rights management], period.

—Jean Bedord, publishing industry analyst at Shore Communications, as quoted in “Have e-books turned a page?” by David Becker on CNET News.com (news.com.com/Have+e-books+turned+a+page%3F/2100-1025_3-5326015.html?tag=st.pop)

Without open standards, software freedom is an illusion. Open standards are the foundation for software freedom.

—Larry Rosen, speaking at Open Source, Open Standards, heard live

The Battle for Wesnoth: www.wesnoth.org

Don Marti

Issue #128, December 2004

The Battle for Wesnoth is a 2-D, turn-based game, with a sword-and-sorcery storyline, slick graphics and music and a lot of levels. Your units gain experience points and power, if you can keep them alive from battle to battle. Occupying villages gives you the ability to heal units and recruit more. Levels seem to make an abrupt jump in difficulty from fighting a few Orcs with allies and an umpteenth-level sorcerer on your side up to trying to defeat two armies of the undead with a handful of footpads and some fish-boys, so it's a good thing you don't have to be good to have some fun with this game.

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

[Advanced search](#)

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

Advanced search

New Products

TextMaker for Zaurus

TextMaker, a word processor available from German company SoftMaker, now is available for the Zaurus platform. TextMaker for Zaurus enables direct document exchanges with other TextMaker platforms without conversion and without lost formatting. Document exchanges also are possible with various versions of Microsoft Word and other file formats, including HTML, RTF, ASCII and Unicode. Other features include a multilanguage spell-checker and hyphenation program, fully customizable keyboard mapping and control strips, an integrated dBase-compatible database and an integrated file manager. TextMaker for Zaurus can run on any Zaurus mode, supports high-resolution screens and landscape mode and can be installed to internal memory or to a memory cards.

SoftMaker Software GmbH, Kronacher Str. 7, D-90427 Nuernberg, Germany, info@softmaker.de, www.softmaker.co.



Appro 1U and 4U Quad Opteron Servers

Appro announced two new servers, the 1U-1142H and the 4U-4148H, built around quad Opteron processors. The 1U-1142H Quad Opteron Server features up to four Opteron processors with simultaneous 32-bit and 64-bit computing capability supporting up to 32GB of ECC 333/400 DDR memory, two IDE, SCSI or SATA HDDs, slim FDD and CD drives or DVD-ROM drive and a full PCI-X 64-bit/133MHz riser slot. The 4U-4148H additionally features eight SATA or SCSI HDDs, up to five PCI-X slots and a 3.25" FDD and 5.25" drive bay for CD/DVD-ROM or tape device. Both servers come with remote server management capabilities and high-speed interconnect options.

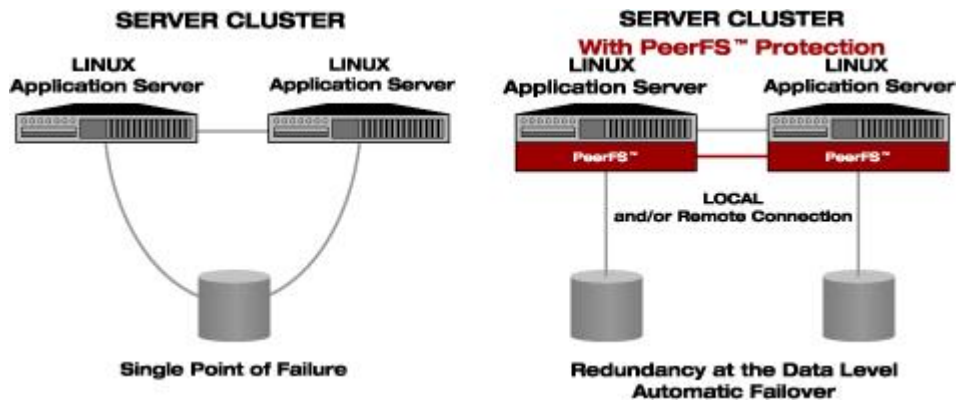
Appro Inc., 446 South Abbott Avenue, Milpitas, California 95035, 800-927-5464, www.appro.com.



PeerFS Data Replication

Radiant Data Corporation released PeerFS, peer-to-peer continuous file replication technology for Linux-based enterprise applications. PeerFS is a POSIX-compatible filesystem that enables data to stay consistent and current across multiple sources and multiple targets. The data always is visible and usable by local applications on any node. PeerFS also enables simultaneous transactions on multiple servers in multiple locations with separate but identical data stores, for applications such as MySQL and other Web services components. Full 256-bit AES encryption is configurable for each endpoint, and PeerFS provides seamless failover among endpoints in the event of storage failures. PeerFS can interface with any application or database and is hardware agnostic.

Radiant Data Corporation, 6273 Monarch Park Place, Niwot, Colorado 80503, 866-652-0870, www.radiantdata.com.



Adonis DNS/DHCP Appliance

The Adonis DNS/DHCP Appliance from BlueCat Networks plugs directly in to corporate networks to integrate DNS and DHCP services for the centralized administration of corporate-wide IP addresses, configurations and hostnames. New features in the latest Adonis appliance include crossover high availability (XHA) architecture to ensure failover for DNS and DHCP services; a lease viewer for managing, in tabular or graphical format, details of IP lease distributions; a data checker for resolving errors before deploying DNS and DHCP configurations; and simplified editing that allows for up to 100 levels of undo/redo with cut/copy/paste functionality.

BlueCat Networks, Inc., 9050 Yonge Street, Suite 401, Richmond Hill, Ontario Canada L4C 9S6, 866-895-6931, bluecatnetworks.com.

BiTMICRO Networks E-Disks

BiTMICRO Networks has introduced a new line of solid state disk-based iSCSI target appliances called E-Disks. E-Disks are available in Fibre Channel, Ultra320 SCSI, PATA and SATA versions. All versions can be used in enterprise installations, Internet communications and traditional industrial applications. The E-Disks are designed to offer a consolidated, interoperable and manageable Flash-based network storage solution that scales over long distances, provides easy connectivity to SANs, allows shared access for multiple users and handles existing user applications with ease.

BiTMICRO Networks, Inc., 45550 Northport Loop East, Fremont, California 94538, 510-743-3475, www.bitmicro.com.



[Advanced search](#)

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

Advanced search

Best of Technical Support

Display Problems on Compaq Laptop

I am using a Compaq Presario 2500, and I have Fedora Core 2 running. The problem is my screen keeps shrinking, but I don't know what is causing it. What do you suggest I do?

—

Anonymous

CC04002@STD.KUKTEM.EDU.MY

There are reports of some problems, as you describe, with ATI RADEON video cards, such as the one your Presario 2500 probably has. One thing worth trying is to comment out the `Load dri` line in the `/etc/X11/xorg.conf` file, and then restart your X session or your computer. You may get unexpected results on your specific machine, though.

—

Felipe Barousse Boué

fbarousse@piensa.com

ACPI, Sleep and Standby

I have an ASUS L8400B laptop, with a 700MHz P-III. I recently installed Fedora Core 2 with kernel 2.6.5. This kernel uses ACPI instead of the APM used by the 2.4 series of kernels, and suspending the laptop works differently and confusingly.

With APM, I used the `apm` command to go into sleep or standby mode. The first, as far as I could tell, kept everything in memory; the second, I think, would save state to disk and then go to sleep. In either case, all I had to do to wake up the laptop was press any key on the keyboard.

With ACPI, I see this in `/sys/power/state`:

Echoing disk into `/sys/power/state` does nothing. Echoing either of the other terms into it suspends the laptop. But, it won't wake up when I bang on the keyboard. If I use the power switch, the laptop does wake up. However, the system also notices that I pressed the power switch, and it starts an immediate shutdown.

Googling has brought me little satisfaction, and the ACPI FAQ seems to be around three years old. So, direct questions are 1) Is suspend to disk supported? If so, how do I make it work? 2) What is the correct way to wake up the laptop? 3) What else do I need to know? I'm a big novice at ACPI.

I would prefer to avoid updating kernels and the like, but I will if doing so would make a difference.

—

Arnold Robbins

arnold@skeeve.com

Suspend to disk on that laptop is a BIOS feature. If you destroyed your suspend partition when you installed Linux, you may need to re-create it. Here's another ASUS owner with some helpful advice: linux.seindal.dk/item20.html.

If you're interested in customizing your ACPI setup, it's easier to debug if you shut down `acpid` by running the appropriate init script with the `stop` argument, then run `acpid` in the foreground with the `-d` option. You can see what ACPI events are being generated and modify the appropriate scripts.

—

Don Marti

info@linuxjournal.com

Daktronics Disses Linux Customers

Why do most manufacturers of devices choose the Microsoft model over the Linux model? Why not develop for both? Specifically, a large manufacturer of LED signs, Daktronics, uses all Microsoft products in the development kit it produces. The company does have a protocol guide but offers no support for the Linux shared library model. I am developing for these signs based on the Linux model, but that work soon will be tanked in favor of Microsoft

development. What can we do to get manufacturers to create libraries to speed up development under Linux? I will continue to develop for this project on my own, as I feel there might be a use in the Open Source community for device drivers for signs such as these. If you can point me toward anyone or a group interested in the same issue, please do so.

—

Chuck Smith

chucksmith@viawest.net

Although advocacy groups certainly are available, such as the OSDL, most vendors respond only to one thing—customer pressure. Add your voice to the chorus, and contact every representative of the company you can to voice your desire for them to embrace Linux. Most manufacturers already are seeing the light in this area, but many find the addition of support for a new operating system challenging, and they need strong encouragement to do so. Patience and consistency are the keys in voicing your concerns. Take heart; over the next two years, it will become increasingly rare for a vendor to disregard Linux.

—

Chad Robinson

chad@lucubration.com

It's great that you want to work on a project, but if you're bound by the license agreement of the proprietary development kit, you may be prohibited from releasing your own open-source code with similar functionality. Check with a knowledgeable law firm such as rosenlaw.com—you're probably better off working on projects in an area where you've never clicked a nasty “I Agree” button.

—

Don Marti

info@linuxjournal.com

Linux Connection for Old Microsoft Exchange?

I'm hoping this question can be answered; I've looked everywhere for a solution and found just about nothing. We're currently trying to do a roll-out of Java Desktop Systems, but we have found that Evolution doesn't support MS

Exchange calendaring and such. To get this working, we found a solution in Ximian Connector, but it doesn't work on Exchange 5.5. Surely, more than only Ximian is out there. Being fairly new to the *nix field, I'm completely clueless as to what else can be used. Do you know of any other software that connects Evolution e-mail clients to an MS Exchange server?

—

Monique Marais

monique.marais@alindigo.com

You have an old version of Exchange, one apparently receiving no further support even from Microsoft. There are plenty of solutions for calendaring and e-mail on Linux; some are free and some proprietary. Take a look at www.calendarhome.com/clink/web-calendar.html for a list of this type of software.

—

Felipe Barousse Boué

fbarousse@piensa.com

You're going to have to upgrade that old version of Microsoft Exchange sooner or later, so consider buying time by installing a VNC client on the new Linux systems, and a VNC server on the Microsoft side, to give users access to your old applications. See the discussion at www.linuxmafia.com/faq/Legacy_Microsoft/vnc-and-similar.html.

—

Don Marti

info@linuxjournal.com

Point-of-Sale System?

I am looking for a good POS system to run a small salon with three employees. I am using Red Hat 9 and would like free projects or low-cost ones.

—

Randy Freer

freers@charter.net

I have evaluated several nice POS systems that may suit your needs. Take a look at www.linux-pos.org, a site that specializes in Linux applications for the retail industry. This page has a large list of free software available for business applications, including POS.

Keep in mind that, depending on your company's needs, you might have to integrate a lot of things, including cash drawers, barcode technologies, ticket printers and so on. Therefore, for practical reasons, comparing the total cost of ownership—including your time or hiring third parties—with a commercial solution, even Linux or open-source ones, is a wise exercise. With a commercial solution, you receive support and a turn-key working solution instead of having to integrate, build and support everything by yourself.

—

Felipe Barousse Boué

fbarousse@piensa.com

Alternate Mail Solution

In reply to Walter's August 2004 Best of Technical Support question: I solved this problem quite a long time ago, see www.trestle.com/linux/trestlemail/bye-trestlemail.html. However, now that I have Postfix handling all my mail, I have no need for TrestleMail. I totally agree with Felipe's answer: as long as you're using multidrop mail, you suffer from mishandled boundary cases (mailing list mail, bccs and so on). There's no way around it. On the other hand, the simple stuff is totally reliable, and you have full control over all mail delivery. You might find that Trestlemail does exactly what you need. Let me know if you have any questions.

—

Scott Bronson

bronson@rinspin.com

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

[Advanced search](#)

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