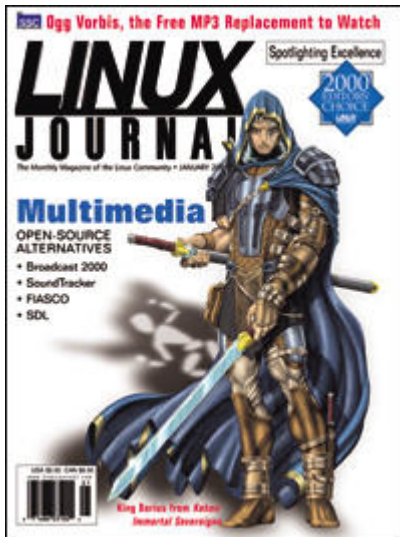


Advanced search

Linux Journal Issue #81/January 2001



Features

Focus: Multimedia by *Don Marti*

Part II: A Profile of SoundTracker by *Dave Phillips*

This article is based on a chapter from the author's book *Linux Music & Sound*, published by No Starch Press in October 2000.

FIASCO—An Open-Source Fractal Image and Sequence Codec by *Dr. Ullrich Hafner*

FIASCO provides state-of-the-art image and video compression.

Moviemaking on a Linux Box? No Way! by *Adam Williams*

Broadcast 2000 aims to bring together the art of making movies and the power of the Linux platform.

Running a Net Radio Station with Open-Source Software by *Andy Faulkner, Rich Smith, Brad Baylor, Jim Bailey, Paul Mack, Jim Lemaster and Tom Hartel*

Seven Linux enthusiasts broadcast a weekly internet radio show.

Streaming Media by *Frank LaMonica*

LaMonica describes the hardware and software technology used on the server side of the streaming process.

The Story of OpenAL by *Bernd Kreimeier*

Kreimeier explores one of Loki's free software projects.

A Crash Course in SDL by *John Hall*

An adaptation of a chapter from the author's upcoming book.

Ogg Vorbis—Open, Free Audio—Set Your Media Free by *Jack Moffitt*

Ogg Vorbis is the Open Source Community's hot alternative to MP3.

An Introduction to MSERV by *Joshua Drake*

Drake explains how MSERV can end musical dictatorship.

Indepth

Code Reviews by *Larry Colen*

The best code reviews are the ones that actually get done.

Toolbox

Kernel Korner Meddling with Memory by *Zhang Yong*

At the Forge Three-Tiered Design by *Reuven M. Lerner*

Cooking with Linux Music to Feed Your Soul by *Marcel Gagné*

Paranoid Penguin The 101 Uses of OpenSSH: Part I by *Mick Bauer*

Columns

Linley on Linux: Home Network Push Accelerates by *Linley Gwennap*

Focus on Software by *David A. Bandel*

Focus on Embedded Systems by *Rick Lehrbaum*

The Last Word by *Stan Kelly-Bootle*

Games Penguins Play: Soldier of Fortune for Linux by *J. Neil Doane*

Linux for Suits The Morlock Market by *Doc Searls*

Reviews

TuxTops Obsidian 30W by *Jon Valesh*

SuSE Linux 7.0 by *Stew Benedict*

StarOffice 5.2 by *Stephanie Black*

CorelDRAW for Linux: f/x and Design by *Clifford Anderson*

Departments

Letters

upFRONT

From the Editor Editors' Choice Awards

Best of Technical Support

New Products

Archive Index

Advanced search

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

Advanced search

Focus: Multimedia

Don Marti

Issue #81, January 2001

Multimedia and the Modern Web

Today, according to Raging Search (see <http://ragingsearch.altavista.com/>), there are only 50,665 web pages containing the words "my cat", but 57,615 instances of "streaming media". This is not the Web we thought we knew. As people put bigger, fancier media on the Net instead of just pictures of Mittens, you might think that we would have learned the lesson of the humble GIF file: don't make human communication depend on a patent license.

But people *didn't* learn it, and we're in for another patent mess with the patented MP3 format and several others. As I write this, I'm checking out the [mp3licensing.com](http://www.mp3licensing.com) page about Broadcasting/Streaming (see www.mp3licensing.com/royalty/broadcast.html). Want to run an Internet radio station with MP3 compression? On January 1, 2001, you'd better get out your checkbook. And "Bob" help you if you want to say something their lawyers don't like.

So what happens now? Do we make some huge corporation in Europe the new Ministry of Information of Internet radio? Hell no. This is *Linux Journal*, and we love freedom. In this issue, we're going to give you a crash course in free, open-source multimedia tools. Remember, when speech depends on software, free speech depends on free software.

SoundTracker and Broadcast 2000: If content is king, low-fidelity content is a mad, drooling king whose nonsensical decrees go unheard while better-sounding invaders march into his country, and the people throw flowers before them. Get SoundTracker and learn how to create MOD audio tracks to keep your listeners happy. Broadcast 2000 lets you edit your own movies with cut-and-paste ease.

FIASCO: Fast, free streaming video depends on a patent-free compression algorithm. FIASCO, which Ullrich Hafner developed over the course of five years for his PhD thesis, is a replacement for MPEG video released under the GPL. Now you don't need to get a patent license to develop video applications.

In “Streaming Media” Frank LaMonica looks at what you need to do to roll out a streaming media site—from codecs to RAID drives to quality of service. And in “Running a Net Radio Station with Open-Source Software”, Andy Faulkner and the rest of the opensourceradio.com crew explain exactly how they did it. They're using the patented MP3 format, though, which brings us to “Ogg Vorbis —Open, Free Audio” by Jack Moffitt. Ogg Vorbis, the free replacement for MP3, is definitely ready for prime time, so vorbize a couple of tracks, clean the wax out of your ears and have a (patent-free) listening session.

Finally, we have some really high-tech articles about writing games that are 3-D both for the eyes and the ears. In “The Story of OpenAL” Bernd Kreimeier teaches us about the new standard API for 3-D audio. I wonder how far you can get in *Heretic2* with headphones on and your eyes closed. And, last but not least, we have a sneak preview of the eagerly anticipated book, *Linux Game Programming*. Learn how to get started in SDL, a fast LGPLed library of game development tools.

—Don Marti, Technical Editor

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

[Advanced search](#)

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

Advanced search

A Profile of SoundTracker

David Phillips

Issue #81, January 2001

The following article is based on a chapter in the author's Linux Music and Sound, published by No Starch Press in October 2000.

SoundTracker is the most feature-packed of all the trackers I've used with Linux. Its interface presents all the program's tools on a single screen, making it easy to understand and a pleasure to use. As you will see, tracking with SoundTracker is a simple process and great fun.

Getting It, Building It

SoundTracker is available in full source code, as well as RPM and tarball binaries. Update patches are also available to apply against previous version source trees. The following instructions detail building SoundTracker from the full source package.

Before building the program, study the Requirements page on the SoundTracker web site and verify that you have all the necessary libraries and development tools. SoundTracker runs in the X Window System, so most of what you will need should be included with any mainstream Linux distribution. However, if you're running an older Linux distribution, you may need to acquire the most recent GNOME and GTK packages to build and utilize all of SoundTracker's features. Visit the the project web sites at <http://www.gnome.org/> and <http://www.gtk.org/> to pick up the latest packages. You should also install Michael Pruett's libaudiofile, available at <http://www.68k.org/~michael/audiofile/>. SoundTracker's installation documentation also specifies what you need to build the program, so be sure to read the INSTALL and README files for the latest instruction updates.

After you have downloaded SoundTracker and acquired the necessary support software, you're ready to begin building SoundTracker. Type **./configure -help** for a list of options you can select to customize the build process, then run **./**

configure (with your selected options) to create the makefiles needed to compile SoundTracker. If no errors were reported by the configuration procedure, you can then type **make** and watch the compilation take place. If no errors were reported by the make process, become the superuser by typing **su root** and entering your root password, then type **make install**. The tracker is now ready to use; you can start it by simply typing **soundtracker** in an xterm window.

Some Preliminary Instructions

As of version 0.5.5, the documentation for SoundTracker consists of a single README file, but little more is needed. Trackers all follow a similar design, and the reference materials found on the United Trackers and MODPlug Central sites will help you understand the basic use of almost any tracker.

If you have never used a tracker, you might want to pick up some mods to load and play in SoundTracker. The resource listings at United Trackers and MODPlug Central will guide you to some superb mod collections. Pick some in either MOD or XM format, load them into SoundTracker, then listen, study and learn.

All sounds used in mods are sampled sounds, so you'll want to build a collection of samples that meets your compositional needs. You can rip samples from existing modules, or you can gather sounds from the various sample collection sites listed on the United Trackers and MODPlug Central web pages. SoundTracker can load any audio file type supported by libaudiofile, so you can freely mix WAV, AIFF and AU files in the same module.

SoundTracker uses only monaural samples. If you try to load a stereo sample, the program politely informs you of that fact, then gives you the option of loading either the left or right channel or a mix of the two.

Spend some time preparing your samples. Tuning and looping are basic considerations, and SoundTracker supports extensive editing of a sample's volume and panning. Although a sample editor is included in the program, you may want a more powerful sound file editor such as MiXViews, Snd or DAP for finer editing. These dedicated editors provide greater resolution, more effects processing and file conversion routines not available in the editors included with trackers.

You may find it helpful to load a number of samples into instrument locations before beginning the tracking process. SoundTracker provides 128 locations for your samples, so you can load large libraries of sounds, then scroll through the Instr number box (just above the Module Info tab) to quickly find the sound you

want. You can easily test any selected sound by pressing one of the “pitched” computer keyboard keys (see below for more details).

Using SoundTracker

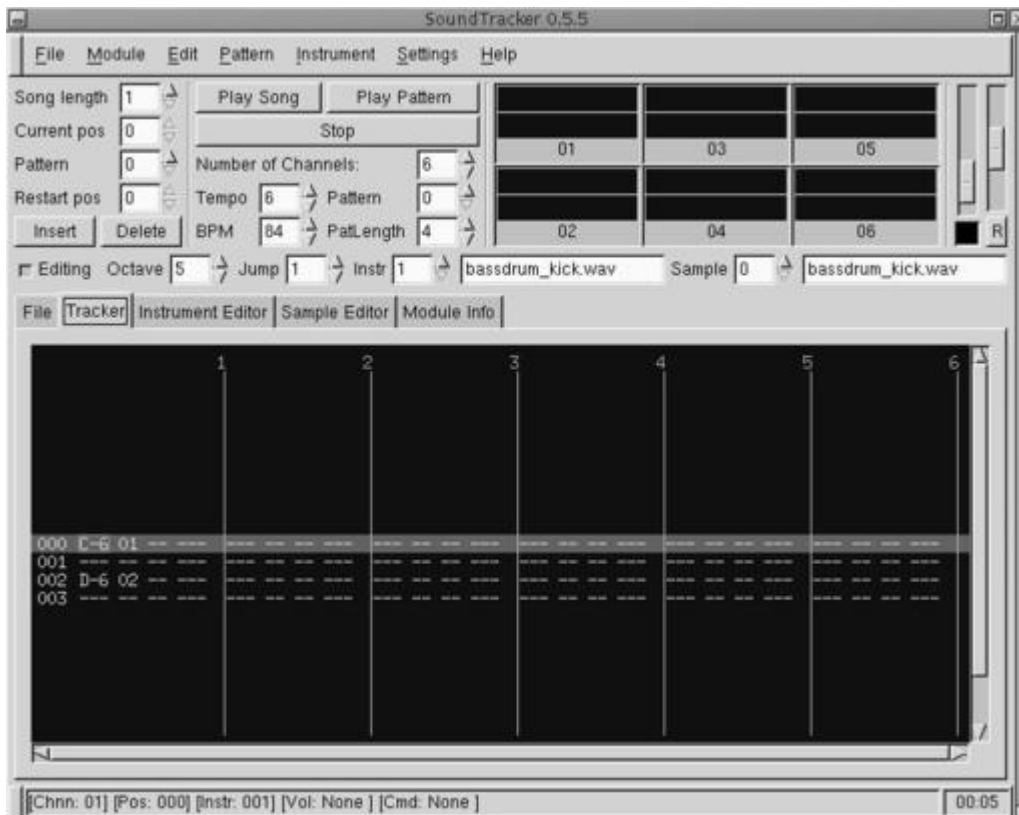


Figure 1. SoundTracker's Tracker Display

Figure 1 shows the basic track display from SoundTracker. I'll discuss how the data got there in a moment, but for now you only need to know that each of the six columns represents a track (also called a channel), and each of the four rows represents a beat.

Here is the breakdown of the rows in the first track in Figure 1:

Beat	Pitch	Instrument	Volume	Effect command	Effect parameters	000	C-6
000	C-6	01					
001							
002	D-6	02					
003							

We see that Instrument #01 (a bass drum sample) plays on the first beat with a pitch of C6, the default value for volume, and with no command information for effects processing. Instrument #02 (a snare drum sample) is played on the third beat with a pitch of D6. If we pressed the Play Pattern button, we would see the track display repeatedly scroll the columns past the rectangular cursor, and we would hear a continuously looping pattern of four beats with bass and snare drums on beats one and three.

Instrument samples are played at the indicated pitches, and the instrument numbers may be changed within the channel. Values can carry or ramp from

beat to beat. The effect command and effect parameter values define the type of effect and its intensity, and effects can be dynamically controlled beyond the first instance of a sound. Note that defining varying pitches for your percussion instruments gives a more realistic sound to drum tracks.

Taken all together, the events in Figure 1 make up a single track in a pattern. Each pattern can be up to 64 beats long and can contain up to 32 tracks. Tracks and patterns can be cut, copied and pasted. By default, all tracks are set to play together. Left click on a track's oscilloscope to toggle the track's mute status, right click the scope to solo the track.

As shown in Figure 1 (SoundTracker's default display mode), the program's global organizing controls are at the top left, a bank of oscilloscopes sits at top right, and the file manager, tracking display, instrument editor and other tools are organized in a tabbed block in the bottom half of the screen. We have seen what a track looks like, now let's find out how to create one of our own.

Entering Events into a Track

You can insert events into a track by using either your computer's QWERTY keyboard or an external MIDI keyboard. Both methods can be used in either real-time mode (entering and deleting events while the pattern loops) or in step-entry mode (inserting and deleting events one by one, manually incrementing the cursor position). Let's look at the QWERTY method first.

Inserting Events with the QWERTY Keyboard

Event entry is easy with a two-octave span of notes mapped to your computer keyboard as is demonstrated in Table 1.

Table 1. Two-Octave Span of Notes

You can change the starting octave in the Octave scrolling window to the right of the Editing button (see Figure 1).

Click on the Editing button, then use the computer keyboard to audition and insert your instruments (samples) into your tracks. Just click Play Pattern while in Editing mode, and you can insert events in real time while the pattern loops (this method is easier and more accurate at slower tempos). Alternately, you can add events one by one by scrolling to an entry point and inserting sounds in step-entry mode. Events are deleted simply by selecting them with the tracker display cursor (or waiting for the event to scroll by in real time) and hitting the Delete key.

You can use the keyboard in other ways as well. Controls are available for song or pattern play, and nearly all the editing functions. The combined keyboard/mouse interface is easy to master, and you'll quickly be able to compose in SoundTracker with ease. (See Table 2 for a list of the keyboard accelerators.)

Table 2. Keyboard Applications

Inserting Events with a MIDI Keyboard

If your Linux sound system utilizes the ALSA drivers, and if you have configured SoundTracker for ALSA support, you can use an external MIDI keyboard to enter events. MIDI input is basically identical to the QWERTY method. However, you'll need to configure SoundTracker for MIDI input.

Open the Settings/MIDI Configuration menu from the top menu bar. Clicking the Volume button lets SoundTracker use the MIDI velocity value for the volume of the inserted event. Clicking on the channel button locks MIDI input to a particular track, so that MIDI Channel 1 equals Track 1, MIDI Channel 2 equals Track 2 and so forth. If you select the MIDI channel button, you must change the MIDI Out channel on your keyboard to input data to another track. If you leave the button unchecked, you can simply use the Tab or Shift-Tab keys to select a track for recording (i.e., you won't need to change your MIDI Out setting on your keyboard). Set the client and port numbers if necessary (the defaults work fine for me).

Tracking a Module

Tracking (i.e., creating a module) in SoundTracker is a simple five-step process:

1. Set your pattern number and length with the controls in the upper-left block of the display.
2. Click on the Editing radio button and select Instrument 1 in the global section.
3. Click the Sample Editor tab to load the sample you want for that instrument.
4. Click the Tracker tab, use the arrow keys to position the cursor box over the first event entry space for Instrument 1 and select the note desired from the QWERTY or MIDI keyboard as described above. Remember, any note can be deleted simply by selecting it again and pressing the computer keyboard's Delete key.
5. Repeat this process for each beat until you have the pattern and instrumentation desired. You can play your pattern at any time, a feature that greatly speeds up the composition process.

Event Details

We have already seen that the event entry line has four fields, shown as columns of numbers, indicating the pitch chosen for the sample playback, the instrument number, the volume and the effects field. The effects field combines the effect command and parameter value into one three-digit entry. Only the pitch indicator follows the traditional pitch/octave notation; all other values are represented by decimal or hex numbering (you can select the representation you prefer).

The effects command not only adds typical effects such as vibrato, tremolo, LFO and filtering, it also provides signals to start the subsequent pattern, jump to a new position in the event list, and even jump to a new position and loop from that point for a specified time. It also provides a fine-tuning parameter for more accurate intonation of a sample at a particular pitch level.

The highlighted line in Figure 2 shows a typical tracker event entry: a bass instrument with a pitch of C6 and the defaults for volume and effects. The entry is located on the first beat of a four-beat pattern.

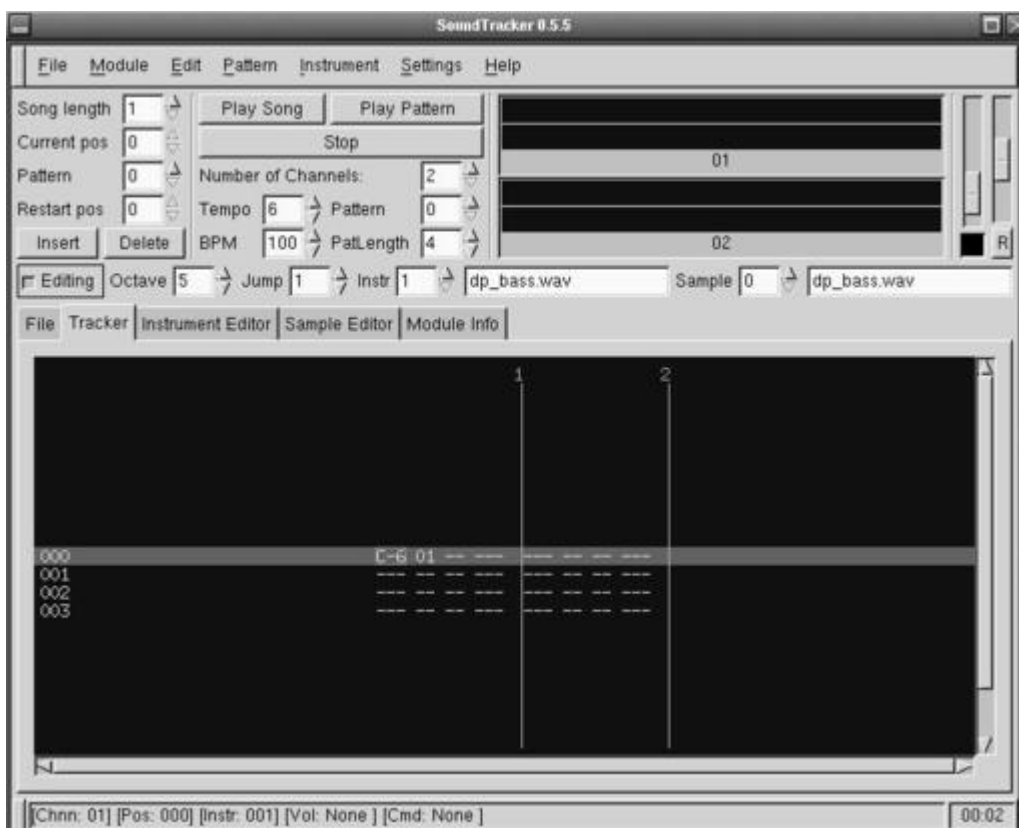


Figure 2. A Single Bass Note in a Four-Beat Pattern

Figure 3 shows the procedure carried out to create a C major scale, rising over a period of 16 beats, from C6 to C7 (the highlight line is mid-scale at F6). In Figures 1 and 2, the tempo has been set to 100BPM (beats per minute), and only two channels are represented, with nothing in channel two yet.

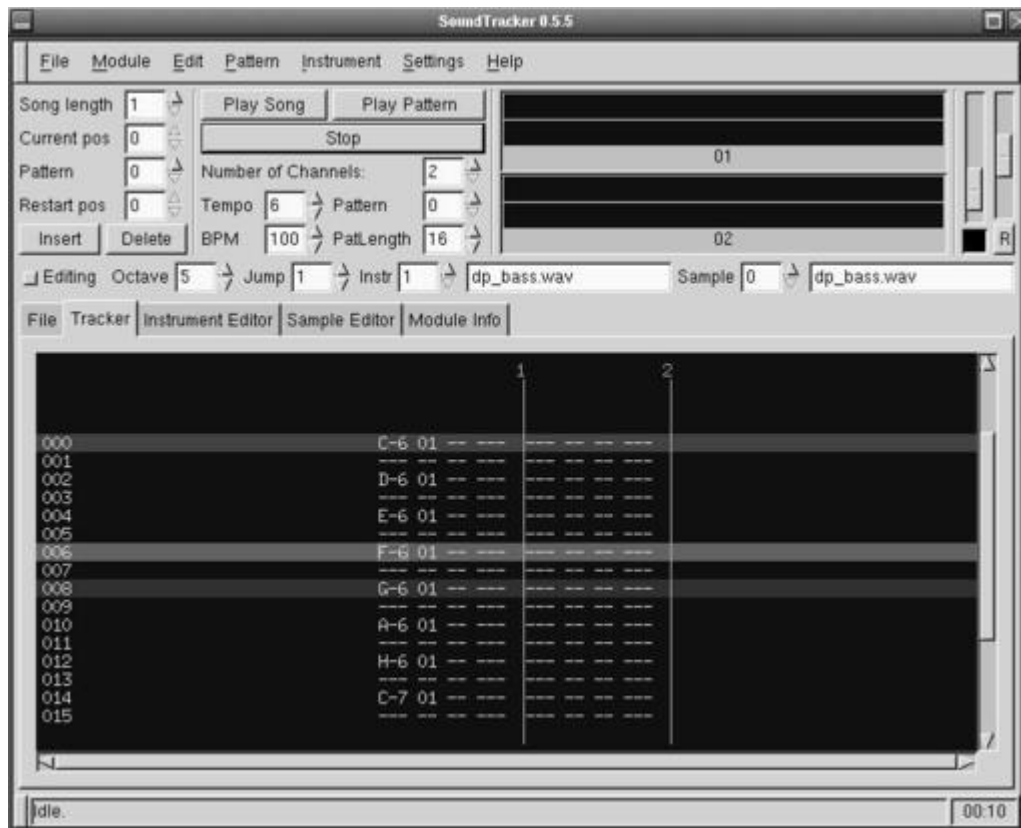


Figure 3. A Scale Played by the Bass Instrument

Note that SoundTracker provides a Tempo control as well as a BPM setting. This control is perhaps best thought of as a kind of “throttle” over the BPM setting. A Tempo of one is the fastest, and all higher numbers gradually slow the speed of the pattern playback. The default Tempo of six matches the BPM to the speed of a metronome with the same BPM setting. Note also that Tempo and BPM are global, so changing their settings will affect the speed of all patterns in your song.

The pattern length and number of channels can be varied at will in the global section. Changing the pattern length or the number of channels in the global section immediately updates both the global and tabbed sections.

Figure 4 shows a more complex pattern from a completed song. The screenshot was taken while looping pattern five, so you can see the activity of the individual instruments in the oscilloscopes at the upper right. The pattern is set to 16 beats (4/4 time) with events added for bass guitar, bass drum, snare, hi-hat, cymbal and guitar.

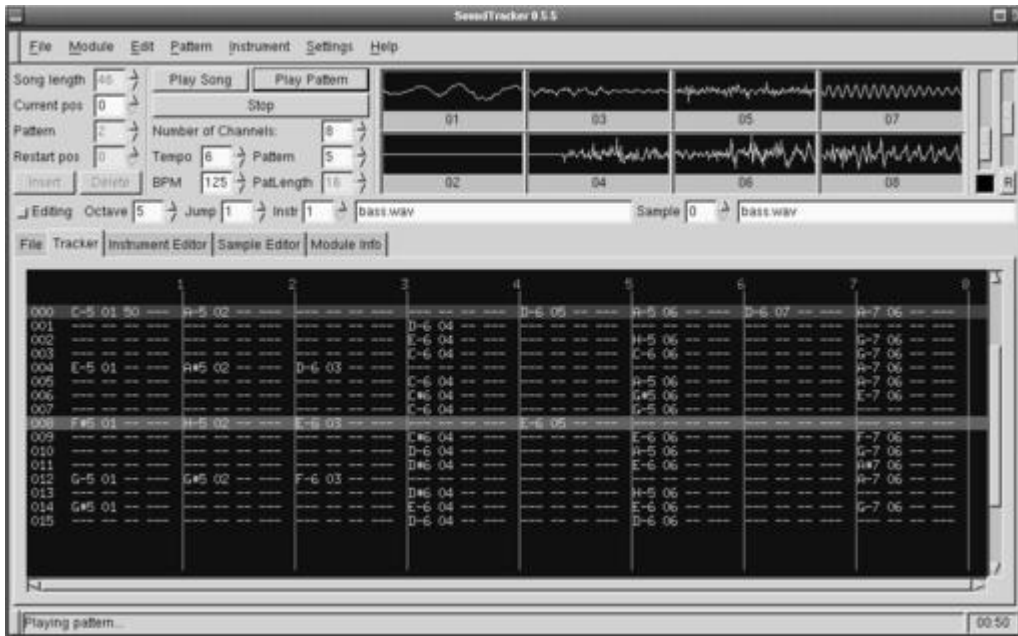


Figure 4. A More Complex Pattern

From Pattern to Song

After creating patterns, you chain them together to create a completed song. This is also an easy procedure. In the upper left corner, set the song length desired (the Insert/Delete buttons can also dynamically control Song length), use the Current Position box to select an insert location point for your pattern, and then use the Pattern box to select which pattern will play at that location. In the example shown in Figure 4, the song length is set to 46 patterns. The display can show only one pattern at a time, but scrolling the Current Position box would reveal that positions 0 through 3 are taken by pattern 2, positions 4 through 6 are occupied by pattern 1, position 7 belongs to pattern 11 and so forth.

When completed, files can be saved in the common XM (Extended Module) format. You can also render your XM to a WAV file directly from SoundTracker.

Additional SoundTracker Features

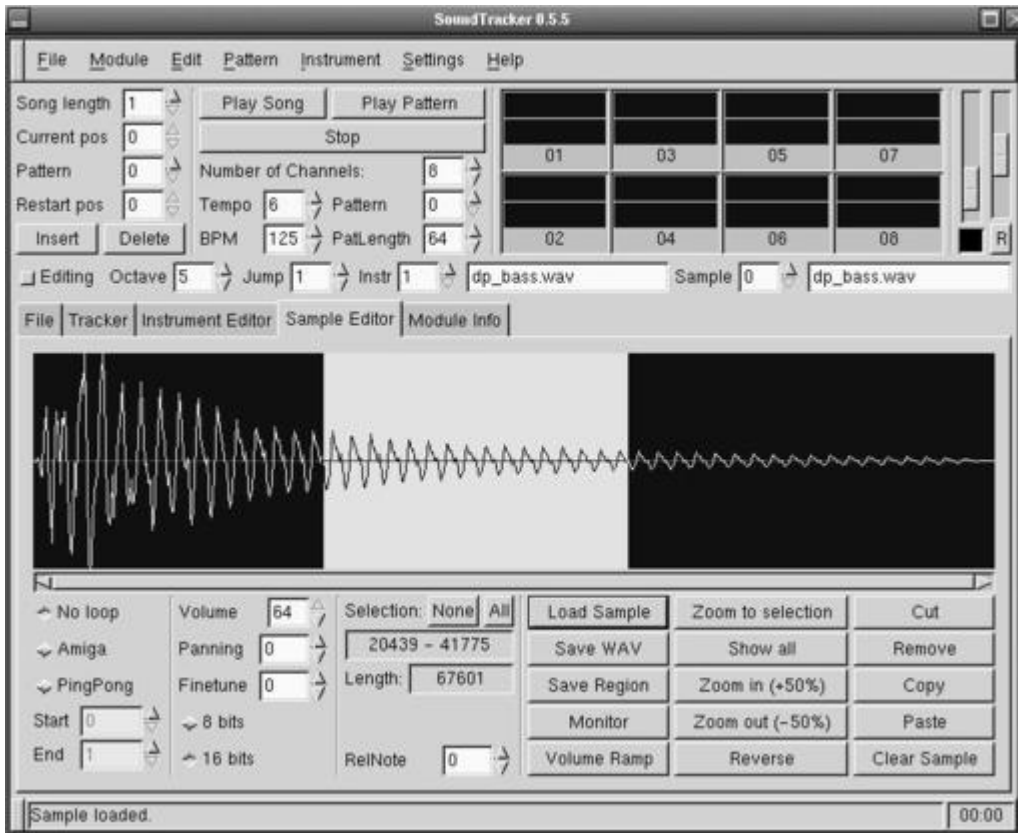


Figure 5. SoundTracker's Sample Editor

Figure 5 shows a bass WAV file loaded in the Sample Editor. SoundTracker's editor is not intended to be a full-fledged sound file editing environment, but it does provide basic cut/copy/paste operations, along with the loop mode settings and initial volume, pan and fine-tune controls. It also has the option to load 8-bit sound files.

You can record your own samples directly into SoundTracker by clicking on the Monitor button. When the Sampling Window appears, click on the Start Sampling button and record away.

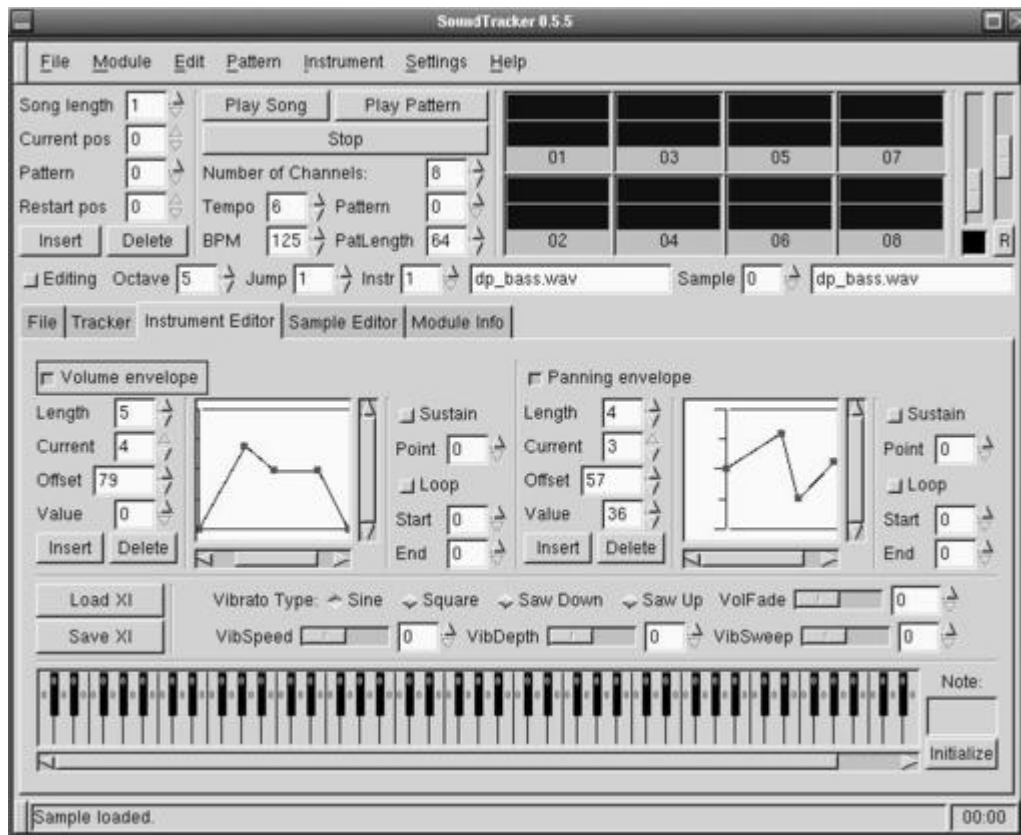


Figure 6. SoundTracker's Instrument Editor

Figure 6 shows the Instrument Editor display where you can edit the volume and pan envelopes of a sample (while the song or pattern is playing, if you wish). This panel also provides a keyboard for mapping your samples across an instrument's range.

XI instruments can be loaded and saved in the Instrument Editor as well as from the Instrument menu in the top menu bar. These instruments are samples specially prepared for use in trackers with defined loop points and envelopes for volume and panning as well as vibrato effect information. You can load and edit your sounds in the Sample Editor, edit them further in the Instrument Editor, and save them as your own XI instruments simply by selecting the Save XI option from either the Instrument Editor tab or the top menu bar Instrument menu.

For more information regarding the XI instrument format, please see the file xi.txt in the SoundTracker doc directory.

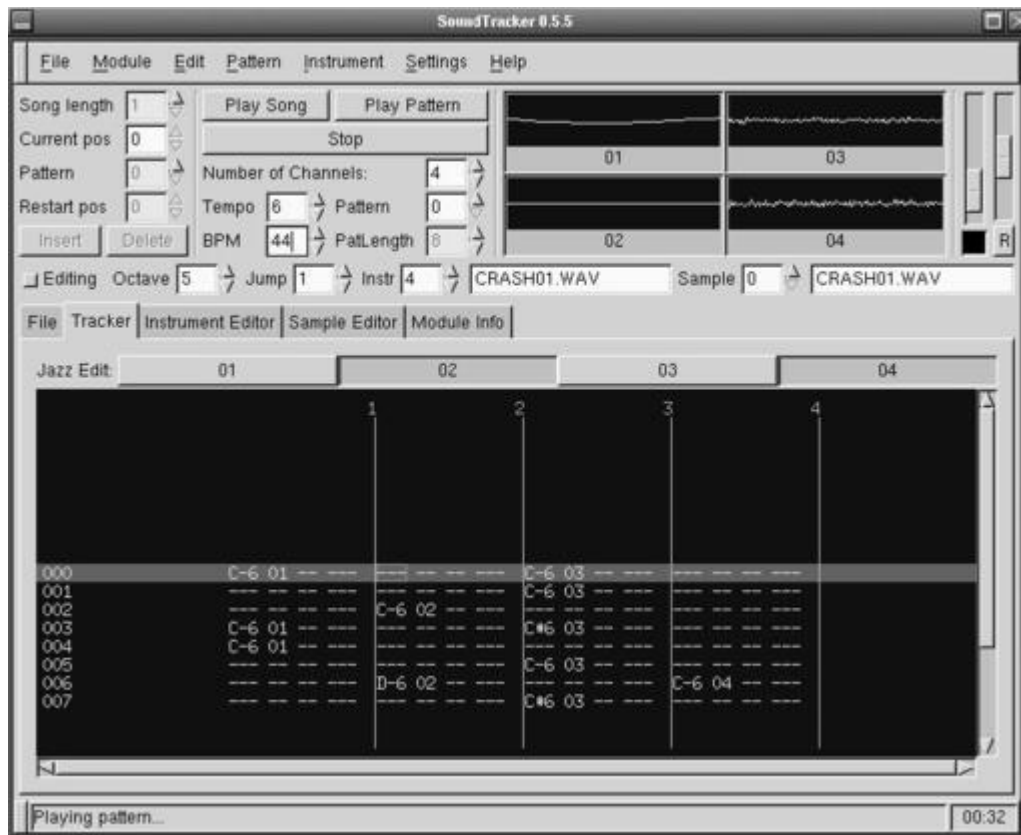


Figure 7. Tracking in Jazz Edit Mode

With the Tracker tab displayed, select Edit/Jazz Edit Mode from the top menu bar. A row of buttons will appear across the top of the Tracker panel, corresponding to the number of channels in your pattern. Click on any of these buttons to define the number of tracks to remain active for Jazz Edit mode. Now, when an event is entered into an active channel, the cursor box will automatically advance to the next active channel.

Figure 7 shows the Tracker display opened in Jazz Edit mode. The pattern has four channels, with channels 2 and 4 selected for Jazz Edit mode. After entering an event into Track 2, the cursor box will advance to the next beat in Track 4; after entering an event in Track 4, the cursor will return to Track 2 and so on. You can override this activity at any time by using the Tab or Shift-Tab keys to relocate the cursor box to another channel.

Jazz Edit is very handy in real-time tracking. In Figure 7, channels 1 and 3 will continue to play as usual while you shuttle between the tracks selected for Jazz Edit mode. Note that you can change your instrument numbers in real time and even load new samples while your pattern plays (though larger samples will likely cause playback to “hiccup”). In fact, real-time edits can be made in any entry box not greyed-out during playback.

You can modify your computer keyboard's note entry behavior by clicking on the Volume Envelope button in the Instrument Editor and then clicking on the

Sustain button (seen at the right side of Volume Envelope display in Figure 6). Now, when you play your keyboard, it will respond like a real synthesizer keyboard, with your “note-off” following the key release. You may need to adjust your audio buffers for the shortest possible latency (open the Settings/ Audio Configuration menu and select Editing Output from the key at the top of the dialog box). In effect, your keyboard can now be played with polyphony (simultaneous sounds) as high as the number of channels selected for Jazz Edit mode.

Closing Remarks

SoundTracker is the most developed Linux tracker available, and it continues to grow in power and flexibility. The program is consistently maintained by its author, often incorporating features and fixes submitted by users corresponding via an active and helpful mail-list. Performance is quite stable, and, as you have seen, the program is very easy to use. Whether you're a seasoned pro or a complete newbie to tracking, SoundTracker is a first-rate tool for creating your Linux mod masterpieces.

The author sends special thanks to Michael Krause and “Mister X aka Kim” for their helpful comments and suggestions. Thanks, guys!

Resources



David Phillips maintains the Linux Music & Sound Applications web site and has been a performing musician for more than 30 years. His work with music software dates back to 1985, and he has been a Linux user since 1995. He is a founding member of the Linux Audio Development group and has been active in the Linux audio software development community since he began using Linux. His publications include contributions to *The Csound Book* (MIT Press, 2000) and several articles in the *Linux Journal*. *The Book of Linux Music & Sound* (No Starch Press, 2000) is his latest publication.

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

[Advanced search](#)

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

[Advanced search](#)

FIASCO—An Open-Source Fractal Image and Sequence

Codec

Dr. . Ullrich Hafner

Issue #81, January 2001

FIASCO provides state-of-the-art image and video compression.

A picture is worth a thousand words—a frequently used sentence to introduce the need for digital image processing. And indeed, a wide variety of aspects in our life is influenced by digital images in the meantime. For instance, in the World Wide Web not only still pictures but also small video sequences are used to enhance the design of web pages. However, the usage of digital images has a major drawback. An enormous amount of data has to be transmitted and stored each time an image or video is requested.

For example, a single uncompressed frame of a high definition television (HDTV) screen (resolution of 1280x720 pixels, 24 bits per pixel) requires more than 2MB memory. When assuming a display rate of 60 frames per second (HDTV), one second of a video movie already requires more than 165MB, summing up to a total of 2,000 compact discs for a movie of 120 minutes! Clearly, downloading such an uncompressed video stream is impossible, even though fast Internet connections like asymmetric digital subscriber line (ADSL) are getting more popular now.

So image and video compression systems—like FIASCO, the fractal image and video codec—are mandatory in handling such enormous amount of data.

Image and Video Compression

Different solutions are applicable to compress image data: for instance, the resolution of the frames can be reduced as well as the frame rate. However,

this reduction is not sufficient. In general, image sequences typically contain three different types of redundancy that can be exploited (see Resources):

- spatial redundancy, which is due to the correlation between neighboring pixels
- spectral redundancy, which is due to the correlation between different color bands (red, green and blue components)
- temporal redundancy, which is due to the correlation between subsequent video frames

The goal of any image compression system is to recognize and remove these redundancies. The following two compression approaches are widely used:

- lossless, or reversible: the decoded image is numerically identical to the original image (the file size is typically reduced by 50%); this is useful if the image is computationally processed any further
- the decoded image contains more or less artifacts (file size less than 10% of the original amount of data); this is useful in low bit-rate applications like the World Wide Web

Hence, lossy compression is mandatory for very low bit-rate Internet video applications (16-64KBps). Several lossy image and video coding standards have emerged in the last ten years, e.g., JPEG, MPEG and H.263. Most of these standards got a face-lift to incorporate the results of the current research in this area, e.g., JPEG2000, MPEG-4 and H.263+ (see Resources). Moreover, a lot of new algorithms have been investigated that have not found a way into a standard, although they are quite appealing. Wavelet-based image coders currently define the state-of-the-art in image coding. However, these codecs still suffer from a slow runtime of the decoder, making real-time (software-based) video decoding nearly impossible. Additionally, most of these new methods are covered by patents and proprietary formats, prohibiting open-source solutions.

Image Compression Algorithms

FIASCO—the fractal image and sequence codec—is intended as a replacement for JPEG and MPEG for very low bit rates (see Resources). It provides the following features:

- state-of-the-art image and video compression (combined in one application)
- real-time software-based decoding
- open-source implementation

FIASCO compressed images are typically much smaller than JPEG files (at low bit rates), while the image quality is still acceptable. For example, see Figure 1 where you see images compressed by JPEG and FIASCO (1:220 in Figure 1A and 1B and 1:100 in Figure 1C and 1D compression ratio, i.e., 0.5% and 1% respectively, of the original file size).



Figure 1A. FIASCO 1:220



Figure 1B. JPEG 1:220



Figure 1C. FIASCO 1:100



Figure 1D. JPEG 1:100

Figure 1. Comparison of JPEG and FIASCO Compression

The coding method of FIASCO is basically a straightforward generalization of the JPEG method. The JPEG algorithm subdivides an image into a set of image blocks of 8x8 pixels. Each of these blocks is then independently encoded by a linear combination of 64 basis images (the cosine basis, see Resources for

details) as shown in Figure 2. The coefficients of these block approximations are then quantized and stored in the output file.

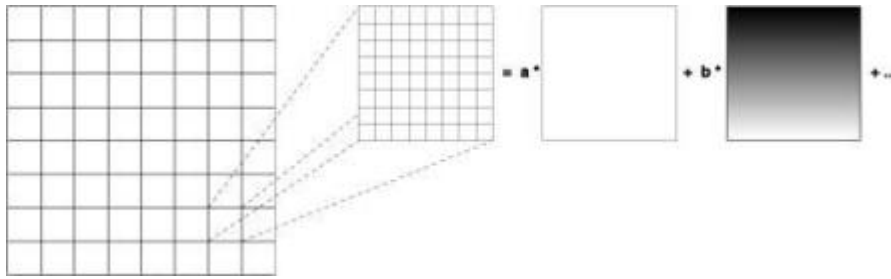


Figure 2. JPEG Compression Algorithm

In Figure 2 the image is subdivided into 8x8 pixel blocks. Each of them is approximated by a linear combination of the cosine basis. The approximation coefficients (a and b in the figure) are then quantized and stored in the JPEG file. FIASCO extends this concept in the following ways:

1. The image is adaptively subdivided into subblocks (squares or rectangles) of different size. Smaller blocks than those used in smooth regions are used in difficult areas.
2. Each image block is approximated by image blocks of a dynamic dictionary. The dictionary not only contains a fixed set of basis images (e.g., the well-known JPEG cosine images) but also all previously encoded image blocks. This method is related to text compression algorithms, which also use dynamically generated dictionaries (e.g., LZW compression).
3. When coding videos, only the difference between subsequent frames is encoded by hierarchical motion compensation.

FIASCO exploits self-similarities within single pictures, and also sequences of pictures, to remove spatial, spectral and temporal redundancies, making it a typical fractal coding method.

FIASCO Source Package

The FIASCO package is released under the terms of the GNU General Public License and is built up of command line applications and a library to compress, decode and visualize images and videos. You just have to run the typical GNU build process (**`./configure;make;make install`**) to compile and install the FIASCO package. FIASCO is entirely written in ANSI C and should compile on most UNIX platforms.

Command Line Applications

Usage of the commandline FIASCO coder and decoder is similar to the **cjpeg** and **djpeg** applications of the IJG package (Independent JPEG group), which should be part of every Linux distribution.

```
cfiasco --quality=10 --output=video.fco frame0*.ppm
```

The above encodes the given video files (in raw PNM format) with quality ten, writing a single FIASCO file **video.fco**. Please note that you cannot compare the quality settings of FIASCO with JPEG. The typical quality range of FIASCO is 1-100, which corresponds to a JPEG quality less than five. FIASCO is intended for very low bit rates only; i.e., you cannot compute compressed images that correspond to a JPEG quality of 75.

dfiasco, the FIASCO decoder, is also used like **djpeg**:

```
dfiasco --output=image.ppm image.fco
```

This decodes the given FIASCO file and writes a new image in PNM format. But **dfiasco** also contains a display module to show video frames in an X11 window. Currently, only an Xlib-based window with the typical set of control buttons is available (play, stop, forward, etc.). Hopefully, this simple decoder and the FIASCO library provides a good starting point for developers to support the FIASCO format in their favorite image and video programs (Mozilla, GIMP, etc.). Please visit the home page of FIASCO to get some compressed images and videos (see Resources).

Compression Library

The coding and decoding functionality of FIASCO is available in a shared library. Data type definitions and function prototypes are provided by the include file **fiasco.h**, which is installed in the **\$prefix/include** directory.

You can simply compress images or videos in your application by a function call:

```
fiasco_coder (image_names, fiasco_name, quality, NULL)
```

This call compresses the image file(s) given by the array **image_names** with the predefined approximation **quality** (1.0-100.0) and creates the new FIASCO output file **fiasco_name**. Several advanced compression parameters can be adjusted by using an optional parameter object which I don't describe here, see the FIASCO manual pages for details.

In order to decode a FIASCO file in your application, you have to instantiate an object of the decoder class **fiasco_decoder_t** with a call of the constructor function **fiasco_decoder_new (fiasco_name, NULL)**. **fiasco_name** specifies the file to decode whereas the second parameter is an optional parameter object to adjust the behavior of the decoder.

Individual frames are then decoded by subsequent calls of the method **fiasco_decoder_get_frame**. These frames are returned in an internal FIASCO format, which then can be rendered in different ways to fit the needs of your application. For more information about the compression library, refer to the manual pages shipped with FIASCO.

Conclusion

FIASCO provides a powerful compression library that is intended to replace JPEG and MPEG for low bit-rate applications. FIASCO is an asymmetric compression method; you get software-based, real-time decoding at the cost of slow encoding times. FIASCO is especially suited in an application context where images or videos are compressed only once but requested and decoded several times (e.g., world wide web applications). Finally, if FIASCO were to be combined with an open-source sound and speech compression (e.g. Vorbis, see Resources), a complete video compression system for low bit rates would be available for free.

Resources

@aa:Ullrich Hafner (hafner@bigfoot.de, ulli.linuxave.net) has been a software engineer in the software management & design AG (sd&m), Germany since 1999. He developed FIASCO for his PhD thesis *Low Bit-Rate Image and Video Coding with Weighted Finite Automats* (see Resources) from 1994—1999.



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

[Advanced search](#)

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

[Advanced search](#)

Movie Making on a Linux Box? No Way!

Adam Williams

Issue #81, January 2001

Broadcast 2000 aims to bring together the art of making movies and the power of the Linux platform.

Years ago there were these HP-UX, IRIX and AIX boxes in the back of college computer labs that did nothing but basic computer infrastructure tasks: file serving, database management, number crunching and maybe a bit of eye candy here and there. If you wanted all of the fun stuff, you had to reboot a Windows PC a few times and wait for the horrendously slow 75Mhz Pentium chip.

But the UNIX boxes were so much faster and efficient than PCs of the time, and a UNIX box editing sound and video seemed like it would transcend all concepts of speed. However, no such software existed. Nothing existed to unify all those efficient utilities and system drivers into a single purpose. Getting a C program that could shuffle gigabytes of sampled audio and video data around in a system, and compile from scratch on thousands of variants of UNIX seemed like the perfect plan.

Windows and Mac advertisements constantly push ways to convert your Windows and Mac workstations into a movie studio. Meanwhile Linux was still pretty much adopted as computer infrastructure glue. If you scour the Internet, however, you might see a few people putting movie studios into a Linux box.

Broadcast 2000 is the Linux movie-making spinoff of mainstream computing in 2000. What Broadcast 2000 does is turn your Linux box into an iMac, a system able to capture, edit and render high-quality multimedia content. The software catapults you into a fury of content sensations normally reserved for those who earn their living by rebooting WinNT.

The Broadcast 2000 software allows you to cut and paste hours of full-motion, full-resolution movies and DVD-quality sound in an instant. The software spits

out 2 gigabyte movie files like poptarts. It exercises the full potential of Linux, and most importantly it crosses the barrier between pure computing infrastructure to productivity applications, which is a challenge akin to landing a human on Mars.

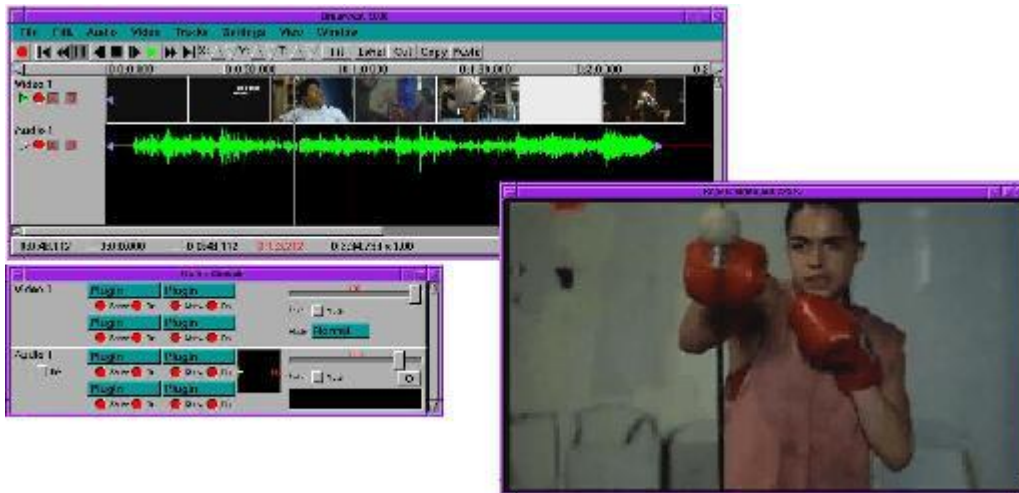


Figure 1. An Internet trailer playing backwards.

Trailers from before 1998 can be transcoded to uncompressed Quicktime by xanim, an exporting edition. Since only the video track has a play icon, only the video plays back. Since each track has a record icon, each track is affected by selections.

It is hard to get programmers who choose Linux based on its tight implementation of basic computer science to focus on movie studio programs. It is even harder to get corporations to support productivity applications in Linux. Impossible tasks like this usually require brute force: sponsors, venture capital and old fashioned full-time programming.

But given enough brute force, Linux boxes can be found recording TV shows, editing commercials, editing home movies, arranging audio CDs, transcoding movies between formats, recording time lapse movies, archiving VHS tapes and playing directories full of MP3s. So which activity is going to enthrall you the most? Probably the installation, right?

Installing It the Easy Way

The preferred method of installing Broadcast 2000 is to download the RPM and invoke

```
rpm -U --nodeps --force <filename>
```

All you have to worry about is matching the following libraries to the point release:


```
XFree86 4.0.1<\n>  
Linux Kernel version 2.2.17.
```

Most problems occur with the RPM command usage. The three options are mandatory since RPM does not cope with the point release fragmentation in the operating system.

E-mails to the Broadcast 2000 support staff usually deal with having an RPM system on the Linux box or not having root access. There is an archaic utility called **rpm2cpio** that substitutes for RPM on these systems. You invoke it in the following sequence:

```
rpm2cpio <rpm file> | cpio -i --make-directories
```

This creates a mini usr tree in your current directory, containing a complete installation. Invoke **ls -lR usr** to view the complete directory tree.

Be sure to relocate the installation to the real **/usr** directory if you have root. Trying to install Broadcast 2000 without root can be very difficult if you are not familiar with runtime libraries. It is worth it to buy your own Linux box so that you can get these problems out of the way.

The script file **usr/local/bcast/bcast2000.sh** contains the proper runtime library path. If you relocated **/usr/local/bcast** you must change BCASTDIR to the new location of **/usr/local/bcast/**.

Installing the Hard Way

Despite the existence of binaries, one-third of the downloads that include Broadcast 2000 from heroinewarrior.com are source code. There are a number of reasons for this: you can learn about optimization strategies by passing different compiler flags; you can also get greater independence by re-enacting as much of the software development process as possible.

Building from scratch is very difficult and time consuming. Most of the libraries required in the executable are statically built by the master Makefile to greatly simplify compiling. In recent years however, many new point releases have emerged, compounding the home-compilation outlook. You need the include files for the following point releases:

```
XFree86 4.0.1.  
Linux Kernel version 2.2.17.
```

This can be done by issuing

```
./configure<\n>  
make  
make install
```

in the top directory. There are undoubtedly going to be problems with newer Linux derivatives that we have not seen yet—and you may need to do some hand coding. If it does not build a point release, mismatch is the likely cause.

Issue the command `/usr/local/bcast/bcast2000.sh` to start Broadcast 2000.

Using Broadcast 2000

Once installed, Broadcast 2000 can perform many operations with a myriad of configurations for each operation. The operations include many alternatives for video capture and audio capture. The system is designed to scale from basic audio to high bandwidth video, but this subjects users to incredibly detailed customization.

Configuration options are described in the `/usr/local/bcast/docs` directory. The biggest help is going to be trial and error until configuration profiles are developed.

Let us look at the three stages to a Broadcast 2000 session: acquiring footage, editing footage and saving a show.

Acquiring Footage

Commonly, e-mails sent to support complain of being unable to load movie files. Most users want to load compressed Internet trailers, but in 1998 the Linux world came to a hard reality. None of the new compression formats were going to run on Linux. Since then we passed three generations of video on the Windows platform, but not a single one was licensed for Linux use.

The only videos that you are likely to play back are MPEG-1, uncompressed Quicktime and DV. Quicktime is not a single compression format. It is really a wrapper for many different video formats. Some of these formats are used in professional studios and some of them are fully supported in Linux. There is also support for loading high-resolution photographs, panning and zooming.

Then of course you can impart footage through video capture interfaces. In a practical sense, there is no footage on the Internet that you want to use in a show, and compression is never used as an intermediate format. When you leave the pure playback domain, all the footage is uncompressed and captured from some esoteric device.

Broadcast 2000 supports Video4Linux, Video4Linux 2, Firewire, LML33 and screencapture interfaces, which are customizable from half duplex audio recording to full duplex video recording. These hardware applications vary in the degree of difficulty of installation. Video capture is usually implemented in

the kernel by patches, separate source code distributions, compilation and hand coding.

Editing footage

After acquiring footage, the biggest problem that people face is editing the interface. Unlike most of the current trends in GUI design, Broadcast 2000 uses a cut and paste strategy. This mechanism allowed precise editing for many years but seems to have shifted out of vogue.

Selecting video is best accomplished by scrubbing: fast-forwarding, rewinding and inserting labels. Then you select the region between the labels by double-clicking on the time bar. Selecting audio can be done just by looking at the waveform and highlighting. The key to success with Broadcast 2000 is the use of labels. Old-timers may remember something called "Soundedit 16", which used the same label paradigm. Here we applied it to video.

Another e-mail that we often see in support is usually "can't isolate tracks for editing". Once again an audio paradigm is applied to video. On old-fashioned 24-track audio decks you had 24 "input/repro" switches. This allowed selected tracks to play back while allowing different tracks to be recorded on. In Broadcast 2000 the ability to isolate tracks for editing and playback is provided by play and record toggles.

On each track there is a play and record toggle that allows just the single track to be affected by cursor selections. Editing is thus purely freestyle with audio tracks being interchangeable with video tracks.

The best part of all is that, when editing, the changes never affect the source files. This makes it possible to relocate hours of footage in an instant, and undo commands are limited to a high 500.

Saving a Show

One of the techniques of persistent storage rarely discussed in Linux circles is the notion of saving a list of pointers to resources already on disk. Broadcast 2000 has two mechanisms of persistent storage: pointer storage and rendering. The "save as" function saves a text file that contain filenames and positions. Professionals call the text file an "edit decision list". Many students flunked out of college trying to get edit decision lists to play on their roommate's computer.

The EDL stores just enough information to reconstruct a show from resources on disk but not enough to transfer across computers. To transfer a show across computers you need to render. Rendering generates a binary file that plays back on platforms.

With a rendered movie, the final stage is compression for the Internet. It turns out that 90% of the production in Internet trailers and clips starts with uncompressed RGB rendered to an uncompressed RGB master and compressed using whatever proprietary compressor is in vogue. In Linux, the final stage would be RealVideo, Mpeg2Movie or LAME. Though not as popular as the proprietary compressor, these Linux compressors actually produce comparable quality.

Recommended Hardware

So you just graduated from MIT and are extremely brilliant. You have vastly greater amounts of disposable income than your peers, and you want to do the impossible. You want to build a Linux box that can produce TV shows.

The ideal target system should be capable of uncompressed, full resolution video playback. A system with lower performance produces a lower quality show even when scaled down to the Internet.

This naturally results in a 45MB/sec hard drive requirement and a 50MB/sec graphics card requirement. Such things are achieved by stringing up several 7,200RPM hard drives on a RAID controller card. At the office we have 75MB/sec through two 10,000RPM drives on a SCSI controller. The RAID controller from Promise achieves 70MB/sec using IDE drives.

Depending on the program length you will need 80GB to 200GB of space. A 200GB IDE RAID gives 107 minutes of play time for under \$700. If you are like my buddy at Micron, however, you will build a stand-alone video server with gigabit Ethernet. The cost of persistent storage is dropping quickly.

Broadcast 2000 uses software to produce effects. Color correcting and compositing 100,000 frames-per-hour of movie requires brute force. Linux software tends to get rated on the minimum CPU and screen size it runs on. In the video world, however, it pays to indulge. Dual CPUs running at greater than 700MHz are recommended. Broadcast 2000 enlists the second CPU so aggressively that you will not be able processor machine after you see dual CPUs in action.

Despite these hardware requirements, the RAM recommendation is only 256MB of 133MHz RAM. RAM is only beneficial when you record video. When you record video, Broadcast 2000 buffers unlimited numbers of frames in memory that defeat hard drive latency. For large RAM systems, however, the operating system tries to cache every disk operation. On our 768MB test system, uncompressed video capture occasionally freezes up for several seconds while the operating system flushes 700 megs of disk writes. Of course, avoid using a swap space too. There are many options for video capture.

Capture | Hardware | Recommended driver<\n> option |
 |-----
 Uncompressed | haupauge | Video4Linux 2analog | WinTV Analog
 |-----Firewire | Generic |
 ieee1394.sourceforge.net-----Compressed |
 LML 33 | linuxmedialabs.comanalog | |

Video playback to a VCR can be achieved with the LML33 hardware.

Each of the drivers has one glitch. You will find that the firewire driver is unable to detect a camcorder without reloading the module set a few times. The Video4Linux 1 driver drops frames. The Video4Linux 2 driver loses sync. And the LML33 swaps field order. But overall a Linux system can be made to generate professional quality video inexpensively with any of the solutions listed above.

The Future

Most of the analog video capture drivers are the result of reverse engineering, persistence and delayed graduations. It was once thought that companies would eventually finance their own Linux drivers or at least document the chip sets. That never happened. This combined with the high cost of video hardware is going to keep Broadcast 2000 from working with most video I/O cards.

The good news is that the new digital TV standards eliminate variations in quality between boards and the need for many drivers with different coding conventions. Each member in the current canon of drivers uses its own coding convention that takes about three months to test and conform to. The current drivers use everything from DMA, FIFO, read/write primitives, stdio primitives, select calls, function overriding and callbacks. This is because the quality of analog video changes for every card. Under the digital system, programmers can have access to the highest quality video by hopefully using only a single coding convention.

The next developments for Broadcast 2000 are updating the interface to future GUI trends, updating the function prototypes as new kernels and C library conventions come out, financing previous development, and hopefully providing new functionality.

Installation is not going to get easier. The installation difficulty arises from many different point releases of libraries being in circulation at any given time. Although aggressive configuration scripts and #ifdefs can temporarily catch up to fragmentation, the real solution is to consolidate the many libraries and sublibraries, consolidate the many kernel options like "Use memory if detected" and preconfigure systems. In October 2000 several companies sold

preconfigured Linux boxes for movie making. One was Linux Media Arts. Others may jump into the art of media creation on the Linux platform.

The trend is now to isolate the Linux interface from user interaction by embedding it in microcontrollers, appliances and computer infrastructure. Part of the justification for Broadcast 2000 is to invent roles for Linux that no one never dreamed of before. It is all about web servers playing movies and routers playing reverb. As a colleague once said on the topic of 3-D webcams for Linux boxes, "That sounds like something only Microsoft would do."



Figure 2. Broadcast 2000 playing a 57 minute WAV file



Figure 3. Traditional Compositing. (Many photos were layered for each scene.)

Resources



Adam Williams is the author of Broadcast 2000. He currently writes software for a company in Silicon Valley, trying to push Linux farther, faster and higher than it has ever gone before.

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

[Advanced search](#)

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

[Advanced search](#)

Running a Net Radio Station With Open-Source Software

Faulkner, Smith, Baylor, Bailey, Mack, Lemasterx, and Hartel

Issue #81, January 2001

Seven Linux enthusiasts decided that it would be cool to broadcast their own radio show over the Internet.

Five great minds got together and decided that broadcasting over the Internet would be cool—plain and simple. Five great minds quickly turned to seven and an Internet radio station at <http://www.opensourceradio.com/> was born. The open-source Community has made this internet radio station possible by supplying all the software components. Two main hardware components are necessary to broadcast on the Internet: a broadcast server and an encoder client. It is not a requirement to separate the server and the client, they can be run on the same machine, but we chose to use two different systems at two different locations to generate our output.

We play MP3-encoded material during our radio show. LAME and Liveice convert the audio output to MP3-encoded specifications so anyone with xmms or Winamp can decode our signal. Broadcasting MP3-encoded material at this time does not require any licensing, however it is anticipated that in the year 2001 royalties will be collected by the agency holding the patent. You can obtain more information regarding these issues at www.mp3licensing.com. If the company that holds the patent imposes royalties, we plan to move to a different encode, decode schema. Ogg Vorbis is a patent and royalty-free compressed audio formatter that we anticipate moving our broadcast server to in the near future. Ogg Vorbis is available at www.xiph.org. Also, the MP3 licensing does not cover issues regarding broadcasting copyrighted material to a public audience so if you plan to run an Internet radio station, be certain you have permission to broadcast anything for which you do not hold the copyright.

We use standard hardware because we found quickly that obscure components introduce unnecessary delays in the setup process. The rest of this article details the setup of each component in our internet radio station. We also describe any problems that we encountered along the way. Please

understand that there are a million ways to create an internet radio station, and the choices we made were not due to any alliance with any vendor—we simply made it work in the following manner.

Figure 1 gives an overview of our network that starts when it gets our voices to an MP3 player. We speak into microphones, the microphones convert our voice to analog and the mixer condenses the stream into the line-in on the encoder client that runs Liveice. Liveice picks up the stream and uses LAME to convert our voice from analog to digital. Liveice then sends our digitized stream out onto the Internet to our defined broadcast server that runs Icecast. Icecast takes the incoming stream and broadcasts it onto the Internet at both the destination and port `www.opensource-radio.com:8000`. Once it is on the Internet, any decoding client, such as an MP3 player, can pick up the stream and decode our digital stream to audio output.

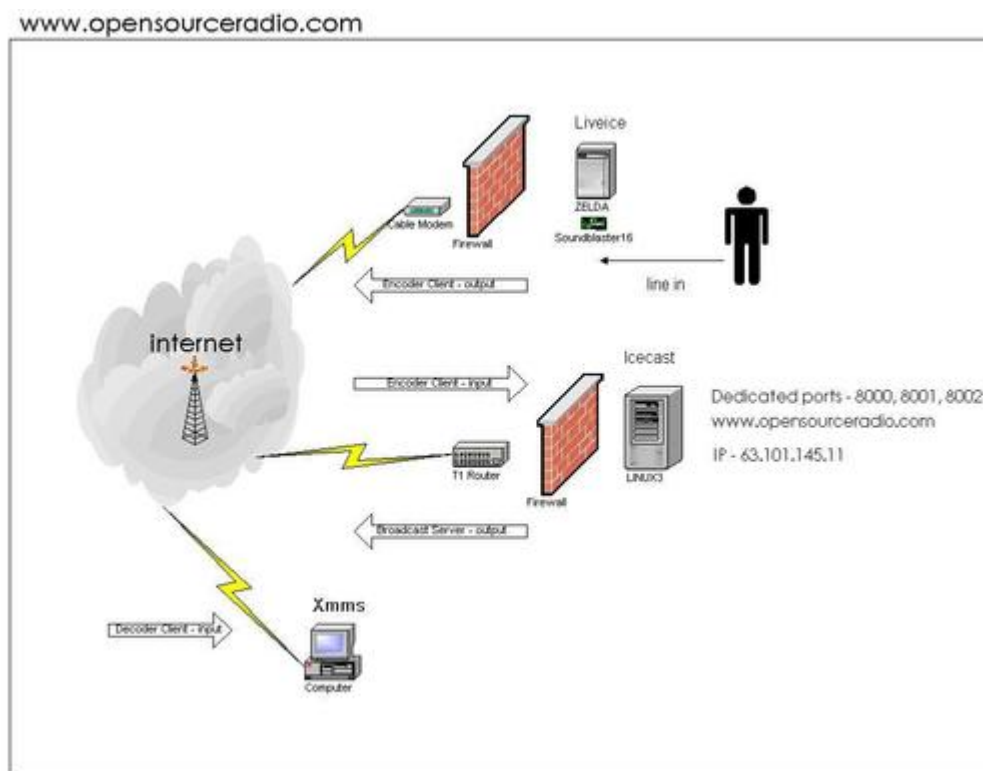


Figure 1. Internet Radio Station Network Overview

Server Requirements

We wanted to succeed right away so we acquired a domain, static IP and a server with open ports. A static IP and domain are not necessary, but by making everything static we don't need to inform our listener base every time our ISP changes our IP. We also connected our server to the Internet via a T1. This gives us the bandwidth to provide high-quality broadcasting. Our broadcast server, Linux3 and our static IP are provided by `www.doitwebcorp.com`. The server is a standard networked PC running Red Hat Linux 6.2.

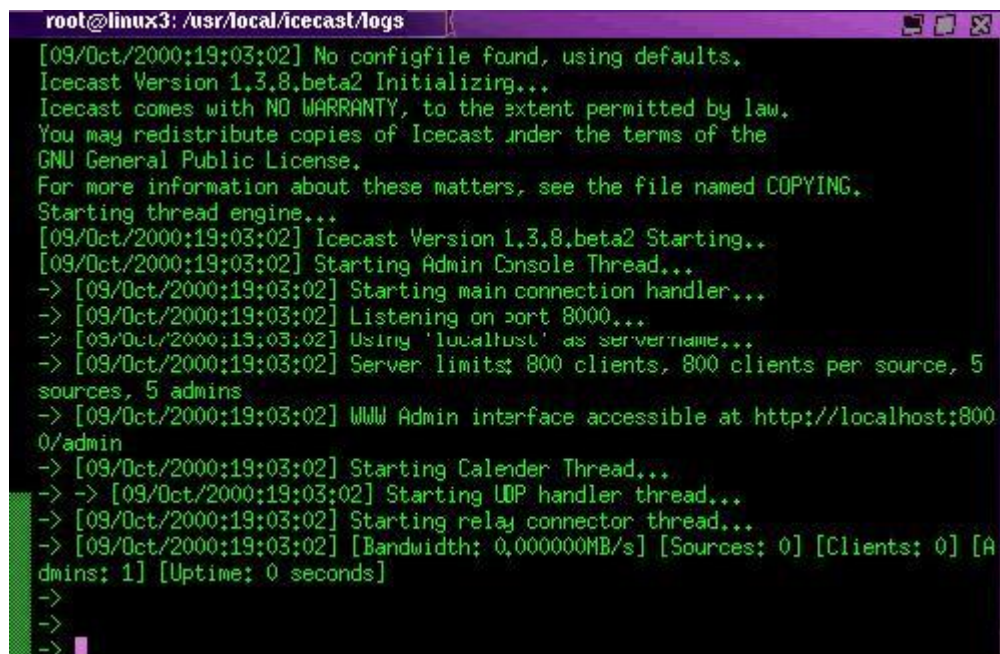
Server Setup

We started by downloading the Icecast server software from www.icecast.org. We chose to use version 1.3.7 because it was considered the most stable release available at the time. We used the default installation that locates the binaries in `/usr/local/icecast`. The configuration of the Icecast server is very simple, you only need to change one file. The `icecast.conf` file is very informative. The sections we changed allow the encoder client to send streaming audio to the broadcast server that will be serving to the Internet. Our server is set up with the IP address 63.101.145.11 with the registered domain of www.opensourceradio.com. If you compare the default `icecast.conf` file to our example `icecast.conf` file, you will see the fields that we changed. This is only a portion of the `icecast.conf` file, as it is a very large file.

Listing 1. Example Configuration File for Icecast

Starting Icecast

Icecast is very simple to run. Go into the account you want your broadcast server to run as and enter `/usr/local/icecast/bin/icecast`. This brings you to console mode on the Icecast server. If you type a `?` at the console you see all of the options that are available with Icecast. We would like to note that from the console you can see when someone connects to the site. If you start Icecast with a `-b` option it pushes the console to the background. Figure 2 is a screen capture of the startup of Icecast.

A screenshot of a terminal window titled "root@linux3: /usr/local/icecast/logs". The terminal displays the output of the Icecast startup process. The text is as follows:

```
[09/Oct/2000:19:03:02] No configfile found, using defaults.
Icecast Version 1.3.8.beta2 Initializing...
Icecast comes with NO WARRANTY, to the extent permitted by law.
You may redistribute copies of Icecast under the terms of the
GNU General Public License.
For more information about these matters, see the file named COPYING.
Starting thread engine...
[09/Oct/2000:19:03:02] Icecast Version 1.3.8.beta2 Starting..
[09/Oct/2000:19:03:02] Starting Admin Console Thread...
-> [09/Oct/2000:19:03:02] Starting main connection handler...
-> [09/Oct/2000:19:03:02] Listening on port 8000...
-> [09/Oct/2000:19:03:02] Using 'localhost' as servername...
-> [09/Oct/2000:19:03:02] Server limits: 800 clients, 800 clients per source, 5
sources, 5 admins
-> [09/Oct/2000:19:03:02] WWW Admin interface accessible at http://localhost:800
0/admin
-> [09/Oct/2000:19:03:02] Starting Calendar Thread...
-> -> [09/Oct/2000:19:03:02] Starting WOP handler thread...
-> [09/Oct/2000:19:03:02] Starting relay connector thread...
-> [09/Oct/2000:19:03:02] [Bandwidth: 0,000000MB/s] [Sources: 0] [Clients: 0] [A
dmins: 1] [Uptime: 0 seconds]
->
->
->
```

Figure 2. Icecast Startup Screen

Icecast Console

The Icecast console is a powerful tool that allows you to control all aspects of the server. For example, if you wish to get rid of particular listeners, you could use the kick command to boot them. Another useful command is dump, which allows you to dump a stream to a file. The full list of commands is available in the Icecast web interface.

Web Administration Interface

Icecast includes a web-based administration interface that can be accessed by entering the URL `http://hostname.domain:port/admin`, where hostname is your Icecast server and port is the port number that you defined in the `icecast.conf` file (see Listing 1). By default the web-based Icecast administration utility is wide open to any system, so be sure to set a password. The help section on the admin page gives detailed descriptions on how to use the web interface. The descriptions include topics that range from setting up user authorization to even disabling the web interface. One of the most useful features of the web interface is a dynamic listing of sources and listener streams. You can configure the administration pages to meet your needs. Refer to the administration interface for more information.

Server Security

Any site on the Internet should be concerned with security. We recommend that you read all the security documentation that comes with the Icecast server. We also recommend that you do not run the Icecast server as root. You should run the server as nobody or a nonprivileged user.

Problems

We did not experience any problems with the Icecast setup and configuration. Icecast itself is very similar to a typical web server, and it is simple to configure.

Encoder Client

Requirements

Our set up uses a remote encoder client that is not located on the same system as our Icecast server. We use the Liveice client software to generate the output to be streamed to the Icecast server. We use a standard workstation, named zelda, running Mandrake Linux 7.2 with a 16-bit SoundBlaster (ES1371) sound card. We chose the SoundBlaster since it is a widely known and easily supported piece of hardware for our purposes.

Encoder Setup

We started the Liveice software installation by downloading the tar file that we found at <http://www.icecast.org/> in the third party applications section. We did not find an RPM at icecast.org. Untar puts the files in our current working directory. Liveice also requires that we run a make to compile the binaries. The README file explains the full installation of Liveice. We put the files in **/usr/local** to make things simple and accessible via the current path. We changed the liveice.cfg file to point to the broadcast server. You can compare our icecast.conf (above) and our liveice.cfg (below) with the defaults to see where you have to make changes. The README file on the Liveice client offers a better understanding of each parameter. The most important parameters include SERVER, PORT, PASSWORD and USE_LAME3.

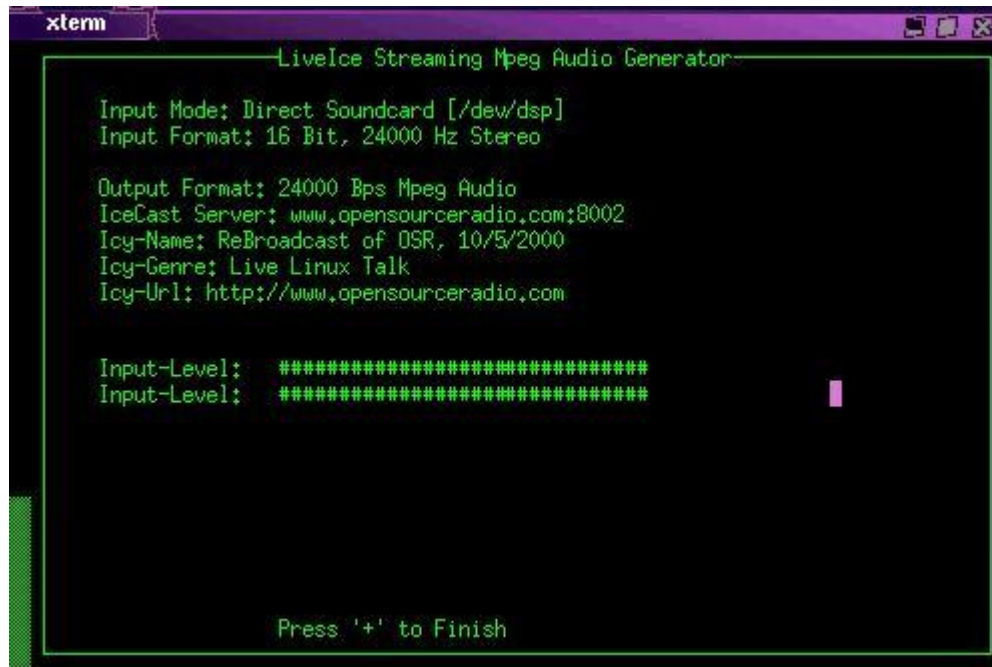
SERVER is the server name that we used setup to the Icecast server. PORT is the port number on the Icecast server. The PASSWORD field must match the Icecast password to allow for the socket connection between Liveice and Icecast. The field USE_LAME3 tells Liveice which encoder to use to convert from analog to digital. The configuration we used works for our purposes, but there are numerous other configurations that accomplish the same output.

```
#####  
# liveice configuration file  
#####  
SERVER www.opensourceradio.com  
PORT 8002  
NAME ReBroadcast of OSR, 10/5/2000  
GENRE Live Linux Talk  
URL www.opensourceradio.com  
PUBLIC 1  
ICY_LOGIN  
SAMPLE_RATE 24000  
STEREO  
SOUNDCARD  
FULL_DUPLEX  
USE_LAME3 lame  
BITRATE 32000  
VBR_QUALITY 1  
NO_MIXER  
PLAYLIST playlist  
DECODER_COMMAND mpg123  
MIX_CONTROL_MANUAL  
CONTROL_FILE mix_command  
TRACK_LOGFILE track.log  
#SAVE_FILE /osr/osr_10_5.mp3
```

LAME is an application that we downloaded from <http://www.sulaco.org/> and we untarred it in the **/usr/bin/** directory. If you put it in a directory that isn't in your \$PATH, you have to put the full path to the encoder in the liveice.cfg file. The encoder is necessary to turn your voice into a digital MP3 format to be received by the Icecast server and broadcast over the Internet. We chose to use LAME, but you can use any encoder engine that you wish to use.

Starting Liveice

To start Liveice, go into the `/usr/local/liveice/bin` and run `liveice`. You must be able to connect to the broadcast server for Liveice to start correctly. Figure 3 is a screen shot of what Liveice looks like when started.

The image shows a terminal window titled 'xterm' with a purple title bar. The main window content is titled 'LiveIce Streaming Mpeg Audio Generator'. It displays the following configuration and status information:

```
Input Mode: Direct Soundcard [/dev/dsp]
Input Format: 16 Bit, 24000 Hz Stereo

Output Format: 24000 Bps Mpeg Audio
IceCast Server: www.opensource-radio.com:8002
Icy-Name: ReBroadcast of OSR, 10/5/2000
Icy-Genre: Live Linux Talk
Icy-Url: http://www.opensource-radio.com

Input-Level: #####
Input-Level: #####
```

At the bottom of the screen, it says 'Press '+' to Finish'. There is a small vertical bar on the right side of the terminal window.

Figure 3. Liveice Startup Screen

Problems

We did not encounter any problems with either the LAME or Liveice setup and configuration. They conform to the open-source standard and are very simple to set up.

Line Input

We purchased an audio mixer that allows multiple microphones to send audio into the line-in port on the sound card. **xqmixer** recognizes the sound card as audio in to send the stream through LAME to be encoded and send output through the Liveice client. After the Liveice client receives the input, the client streams the output over the Internet to our Icecast server. Icecast receives the stream from Liveice and relays it back to the Internet in an MP3 format to be interpreted by any MP3 decoder client that our listeners choose to use. **Xmms** and **Winamp** seem to be the most popular programs to decode MP3 streams.

The mixer provides a greater range of input options. We can run a CDROM, microphones or an MP3 player right into the mixer and out to the Internet. The open-source radio show runs a full range of input to give the true radio feel when we broadcast. We use an audio mixer with six ports for input. Each host has his own microphone that jacks into the mixer, which then connects into the

line-in port of the sound card. Any device that generates audio output can be plugged into the line-in port on the sound card.

Problems

Liveice requires specific audio input quality. The sound card, while it receives input, is still controlled by the tools within the operating system. In our case we use xqmixer to control the sound card hardware. On xqmixer, the record volume controls the streaming rate that Liveice receives as input. If the record volume is set too low, you will not hear any output. If it is set too high, Liveice clips the audio. Clipping is an audio engineering term that describes what happens when the audio signal is too strong for the hardware to handle—it “clips” off part of the signal, making it sound terrible. We adjusted our sound quality by tuning the record volume. It's simple: fire up your station and listen to yourself. If you do not hear anything, increase the record volume. If the playout is clipping, reduce the record volume until you get it right.

Conclusion

The <http://www.opensourceradio.com/> show runs every Thursday night from 8:00 P.M. to 10:00 P.M. EST. We discuss open-source issues and use the full range of our mixing and MP3 conversion capabilities. Anyone can do what we have done. With the exception of the mixer, the computer hardware and the T1 connection, everything was free. You can easily download all of the software from the Internet to create an Internet radio station. We would also like to say that despite our sound, no small animals are injured during the course of our broadcast.

Resources



The hosts of open-source radio are professional engineers by trade and hackers by night. The show they broadcast reflects what they run into every day in the high-tech industry. Andy, Rich, Brad, Paul, Tom, Jim and Jim have all worked to make this dream a reality. You can reach open-source at www.opensourceradio.com or at dj@opensourceradio.com

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

[Advanced search](#)

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

[Advanced search](#)

Streaming Media

Frank LaMonica

Issue #81, January 2001

Frank describes the hardware and software technology used on the server side of the streaming process.

Modern technology is not only growing fast, but the growth literally is exploding. There are so many areas of specialized technology that it has become impossible for one person to fully comprehend the intricacies of the subsystems that combine to produce the latest buzzword. Just as we now use acronyms inside acronyms to succinctly describe the latest innovation, we encapsulate hundreds of human years of work into a single abstract expression, and then piece the abstractions together to create larger, more complex systems, which themselves soon become just another piece of a larger puzzle.

A logical question to ask is “does anyone need to understand systems at that level of detail?” Can't we just piece together all the black boxes knowing only their input/output (I/O) specifications and achieve a working solution? The answer is clearly no. We don't need to understand the most minute detail of every technology we use, but if we are going to make the best use of our technology, we must look deeper than just the visible specifications of each subsystem.

Multimedia content delivery is a good example of a task that requires this type of detailed understanding. Multimedia is simply a combination of two or more types of communications media, such as audio, video, graphics, text or any other element that can stimulate the human perceptual senses. This article looks at streaming media—the process of using computer-based technology for time-dependent delivery of multimedia data as opposed to time-independent data delivery—to show how a fairly deep understanding of many complex technologies must be achieved in order to make these subsystems function together to achieve the desired result. The steps outlined in the streaming process can be presented linearly and, as such, might appear to be

independent black boxes. However, the ideal implementation of each step requires knowledge and a consideration of all of the steps in the process.

Overview of a Streaming Media System

The streaming media process is simply a way to communicate data. This communication can originate from many sources, and it can be targeted to many destinations. The target of these communications is called the receiver. Each receiver can also be the source of data for other “downstream” receivers. The four permutations of source/destination streaming media flow are: One to One, One to Many, Many to One and Many to Many. Each of these four combinations requires a set of common streaming media services, colored by the specific requirements of the source and destination mix. Protocols have been developed by the Internet Engineering Task Force (IETF) to enable all aspects of streaming media, and the industry is only beginning to fully implement these designs.

On the hardware side, a system should be designed with growth and scalability in mind. A relatively small up-front investment into higher performing components could provide large savings later on. It is also very easy to waste money upgrading components that do not affect the areas in the systems where performance bottlenecks actually exist. We will follow the logical path from data creation to transmission to the receiver and discuss the details as we go.

Data Creation

There is an entire industry devoted to the task of creating multimedia data. Historically, expensive, highly proprietary workstations create most high-end, “Hollywood grade” multimedia. The PC revolution produced lower-cost alternatives for many of the tasks that previously required expensive solutions. These solutions were initially on Macintosh computers and have migrated to Windows-based systems, but they were still proprietary systems that locked the user into a dependency on the vendor.

There has recently been a move to Linux. Besides the “free beer” aspect of Linux, end users see a real advantage with Linux because they can free themselves from a dependency on any single vendor. Time-critical content creation is the hallmark of Hollywood production, so companies cannot depend on even the most propitious vendor. In order to protect their own businesses, they must be in a position to do the whole job themselves.

The Visual Effects Society is an industry group with about 24 member companies, each of which base their business on performing the various tasks required to create multimedia data. That society has announced a desire to

move entirely to Linux within the next few years, but there is much work that needs to be done on the Linux OS in order to make that possible. Fortunately, the migration of streaming technology to the Linux OS is well underway.

If you want to provide streaming media services, the entire process depends on the data that you intend to stream. Will you stream live video? DVD or CD-ROM content? Will the data include computer-generated graphics images or some composition of several media types? How will the data be mixed and manipulated in order to produce the final data that you intend to stream to your clients? These questions must first be answered at a strategic level—what is your planned business? Then the answers must be researched based on the equipment, skill sets, time and financial resources available. Those parameters must be considered in terms not only of the data creation itself, but also in terms of the remaining services that you must provide for the entire solution. If, for example, your business is to stream recorded content to paying clients, you had better understand the requirements of your target audience. Will they pay per view, or will they expect “free” content? What is the mix of client technology that you have to support with your streams? These, and many other questions, which we touch on as we explore each step in the process, affect your decision on the software and hardware you need in your system.

Data Encoding

What is data encoding and why is this necessary? The most important process of encoding for computer delivery requires that the data be put into digital form. The real world is based on continuous or analog data. Most computers in use today are based on digital technology. Everything in the system must be represented by some combination of 1s and 0s because all any computer can do is turn individual circuits on (1) or off (0).

Two other factors exist that require encoding to be more than just a simple conversion of analog to digital data. Because typical data is owned and current majority opinion is that it must be protected from illegal copying and distribution, the digitized data stream is most often encrypted during or after the encoding process. Technological limitations on the speed of data flow over various communications links also require that the data be reduced in size, or compressed, so that existing hardware can smoothly execute the streaming process in real time.

The method you use to encode your content depends on encoding software and hardware, CPU processing capabilities and speed, network technology, and the data storage and retrieval requirements. If you intend to stream the data to an audience under your direct control, your questions will focus more on the technology required. If your business is to stream arbitrary data to arbitrary

clients, in addition to the technology, you have to consider standards, de facto standards and specific customer requirements.

The term **codec** refers to the coder/decoder pair that is used to encode and decode your content. Most codecs are proprietary, and because of the huge volume of data that has been encoded in these hidden formats, it is likely to be some time before the industry completely moves to open standards, if it ever does. The most popular proprietary codecs used today come from RealNetworks, Microsoft and Apple.

However there is a strong movement toward the Moving Picture Experts Group (MPEG) formats. MPEG formats have the advantage of being recognized as international standards, and they offer the highest compression ratios with the smallest loss of data quality. There are two versions in common use, MPEG-1 and MPEG-2, but MPEG-4, a new format designed specifically for interactive multimedia applications, is likely to become the standard codec until MPEG-7 is released (possibly in 2001). The most popular flavor, MPEG-2, contains patented IP that can be used by executing a single MPEG-2 patent portfolio license (in the US, <http://www.mpegla.com/>).

Although the licensing is not free, the encoding and decoding algorithms are open, and open-source implementations can be created. MPEG LA announced in September of this year, a plan "aimed at providing fair, reasonable, nondiscriminatory worldwide access under one license to patents that are essential for implementing the international MPEG-4 Systems standard". Use of open standards, such as MPEG, solve at least one main concern of companies encoding their multimedia technology. They will not be forced to return to any specific vendor for service in the future. There are so many patents that cover MPEG technology that it will take a significant engineering breakthrough to produce a competitive freely licensable alternative.

Do you need to understand the technology behind each codec thoroughly before making a decision on which one(s) to use? I would say that a complete understanding is not necessary, but certainly you should understand to the degree necessary to estimate resulting file sizes and to take into account any encoding and decoding overhead. You should also understand the needs of your clients before you decide on the codecs you will use. Proprietary codecs often have no client support for any platform other than the initial encoding platform. It is also very important to consider any technology licensing issues, and be especially conscious of how you will deal with financial liabilities that you may accumulate because of the use of specific technologies.

Data Storage, Retrieval and Transmission

Once you have required hardware and software to create encoded content, consider how you will store your content for transmission at a rate that satisfies the demands of your clients. If your data comes from a real-time, direct-media feed, such as from a video camera, current encoding hardware usually allows only one live feed per encoding card, and your storage requirements are simply the system RAM needed to hold encoded data before it passes to the Network Interface Card (NIC). For One to One, or One to Many data flow, this type of content delivery is straightforward. Because each live feed you supply to your clients requires an individual encoding card, and usually a dedicated computer as the server for that stream, use simple addition to figure out the increased costs.

If your data is preprocessed and streamed either at some later scheduled streaming time, or video on demand (VOD) by the client, then the hardware and software you use for these functions could become a critical bottleneck to the data flow through your system. Here is an example to demonstrate the type of information you will need in order to estimate your system resources.

Suppose your business expects to supply a maximum of 10,000 clients each with continuous stream for one hour, but with a peak time of two hours per day where 80% of the clients could be expected to be on-line. Suppose that the average client demands 300 KBps streams to display flicker free, uninterrupted multimedia content. Suppose also that on average there are 100 different content files being accessed at any time from a pool of 5,000 titles.

Looking at the peak demand sets the upper-limit requirements of our data storage and retrieval subsystem. We decide not to offer VOD to our clients, but allow them only to tune in to some scheduled broadcast. That strategic decision allows us to weigh the number of different pieces of simultaneous content much more heavily than the maximum number of clients. Assuming that your network router can handle multicasting, i.e., sending out the same information to as many IP addresses as may register to receive the data (multicasting is just beginning to become available through some local ISPs), we have enough information to estimate the storage and retrieval subsystem requirements. Our total storage requirement is the product of the number of titles, the average run length of each file and the bit rate at which the data will be streamed. In this case, it is $5,000 \text{ files} \times 3,600 \text{ seconds run length per file} \times 300,000 \text{ bps} / 8 \text{ bits per byte of data} = 675 \text{GBs of storage}$.

How fast must our data travel from the disk drive to the NIC in order to keep up with the expected client demand? To calculate this answer, we compute the product of the number of different files being read simultaneously from the disks and the average data rate: $100 \times 300,000 / 8 = 3.75 \text{MBs per second}$. Had

we decided to offer VOD, that number could jump by a factor of 8,000, the peak number of users who would be asking to start viewing the stream at totally random times, to give us a requirement of 30GBs per second of data needed to be read from our disk drives! We would most likely want to reduce that bandwidth requirement by some intelligent management method. For example, we could choose to allow a new video to start only at the start of each minute. That would be fairly transparent to the end user but would help by allowing us to take advantage of multicast capabilities in our system. We now have to consider the size and number of disk drives, the maximum average data-transfer rate from the drives, and the maximum data-handling capacity of the PCI bus where the data must pass twice before it gets streamed (once from the disk drive to the host, then across the bus again to the NIC).

We cannot find one disk drive that stores that much data, so we have to come up with a scheme to divide the storage across at least several drives. We also know that our clients will be really upset if their show dies midstream, so we have to create for some type of backup plan to account for possible disk failures. Solutions to these problems are often handled with RAID systems and with sophisticated load-balancing software.

If a RAID 0 solution is used, where two identical drives contain mirror images of the same data, how do we access the data in the most efficient manner? Each disk controller typically contains proprietary control software that attempts to optimize the data flow from multiple disk drives. This is a very complex problem, and there is no universal "best" solution. If only two files are being accessed, we can read one file from disk one while we seek the correct track to read the next file from disk two. Now we add a request for file three. Is it more efficient to get it from disk one or from disk two? Maybe it would be best to interweave sectors from each of the two disks, but remember that the slowest operation on a disk drive is the track seek time.

We can help tune our system by some intelligent management of file placement. We might load popular files on all of the disks in our system, but only load the less actively viewed content on two different disk drives. That approach assists in load balancing and reduces the total storage required for necessary redundancy.

The more we know about how our disks and disk controllers operate, the better we can tune our system for optimum performance. Here again, we do not need to know exactly how our storage and retrieval subsystems works, but the more we know, the better off we are.

The final link between your system and the Internet is provided by your network cards. Try to engineer your system so that multimedia content has the

shortest possible path to the Internet. Be conscious of the total bandwidth limitations in your network and in each NIC, and remember that adding multiple NICs to a single computer may not significantly improve your throughput due to bus speed limitations.

Theoretical maximum throughputs listed in specifications are rarely achieved in the real world. When trying to estimate the amount of hardware you need in your system, good engineers who know how to find and benchmark bottlenecks are worth their weight in gold. Before you purchase a huge system, build a smaller prototype and spend a little time finding out where the bottlenecks lie. The better you profile your system, the more effective you will be in tuning it to handle your cost requirements.

Data Streaming

Although the hardware you choose for streaming media is similar to that which is required for any LAN or WAN data transmission, the software determines how efficiently that hardware is used, and, ultimately, what hardware you should purchase. Apple provides an open-source version of their streaming server called the DARWIN Streaming Server. It requires the execution of the Apple Public Source License (see Resources). The DARWIN Streaming Server can stream "QuickTime Hinted" files, which are proprietary to Apple, but because the server is open source, it is likely that it can be modified to support other file formats.

QuickTime is not currently supported on the client side under Linux. Lucent Technologies recently announced the free availability of its OptiMedia MediaServe streaming media server application for Linux (see Resources). They claim to use the industry-standard Real-Time Streaming Protocol (RTSP) and to support a variety of file formats that should make the server useful across many client platforms.

All the other streaming servers I have reviewed are closed source. Some, like Entera and RealNetworks, run on Linux (see Resources). Entera's TeraCAST and TeraEDGE streaming servers use open standard RTP/RTCP streaming protocols in conjunction with RTSP. RealNetworks has its own proprietary RDP protocol that they use with RTSP to communicate to RealPlayer clients. In contrast, Microsoft's proprietary technology, has not (yet?) been ported to Linux.

Quality of Service (QoS)

Let's review some basic streaming technologies that are usually mentioned glibly as acronyms. What are they and how do they interact with each other? The ultimate goal is to introduce you to QoS, the Internet term that refers to the performance, reliability and adherence to any real-time requirements and

any other aspect of Internet service that could impact communication between the source and destination. Many of the commercial streaming-media products differentiate themselves in the market by their QoS capabilities. Be careful. You will be confronted with many decisions on hardware and software purchases based on the level of QoS you want to provide to your expected audience, but unless you control the complete end-to-end path from source to destination, you will always be subject to QoS provided by the poorest performing link in the path. All you can do is make sure that your piece of the path is the best it can be and that your server software makes optimum use of existing QoS protocols.

- IP (Internet Protocol) is the most basic protocol used over the Internet. According to the DOD STANDARD INTERNET PROTOCOL, 1990, IP provides "...the functions necessary to deliver a package of bits (an Internet datagram) from a source to a destination over an interconnected system of networks". All the other streaming protocols and mechanisms sit on top of IP.
- UDP (Unreliable Datagram Protocol) is the usual choice of packet types to send over IP if it is not critical that the data arrive at the destination. This is a low overhead protocol that is especially useful for RTP.
- RTP (Real-Time Protocol) is a real-time transport protocol that sits on top of UDP because it cares much more about "letting the show go on" than sitting around waiting for perfect data. RTP was originally designed to support multicast-multimedia conferencing, but it can be used for other applications.
- TCP (Transmission Control Protocol) on the other hand, has been engineered to provide robust, error-free data transmission. It operates at the same level as RTP but is used in most time-insensitive Internet communications or connections that require high reliability of data.
- RTSP is an application-level protocol that uses the lower-level elements of RTP to manage multiple streams that must be combined to create a multimedia session. These streams can originate from many sources, even geographically detached locations.
- RTCP (Real-Time Control Protocol) packets are carried over TCP connections to work in conjunction with RTP packets to monitor QoS and present feedback to the participants in an RTP session so they can take any actions possible to improve the QoS.
- RSVP (Resource ReSerVation Protocol) is closely tied to QoS. RSVP has been designed to work as a separate process that allows an application to request various QoS levels. Nodes along the route to the requested data analyze the RSVP requests, decide if the requester has the rights to that QoS level and that the requested QoS level is available, and either report back that they will provide the requested QoS or report an error.

You do not need to completely understand any of these protocols, but you should know that RSVP is an extremely rich protocol that gives server vendors a lot of room for QoS improvement. Before you choose your streaming-media server software, it would be useful to ask how their product handles QoS issues, especially via the RSVP protocol.

Decoding and Viewing

These functions are performed on the client computer, but I will mention them briefly here in respect to how they could affect server-side systems. Because of the nature of standard Internet protocols, it should not be necessary to consider the client platform when deciding on the server system. Unfortunately, the power of an illegal monopoly can ignore open standards and force its own proprietary technology on the market. As the open-source movement grows, this type of control will be more difficult to sustain. If you plan to stream closed-proprietary data formats, then your choices on the server side are severely limited. You might want to weigh the future value of promoting a process that restricts competition before you make business decisions that force you down that road.

Conclusion

The need for streaming multimedia content over a LAN or WAN has created a huge market for every conceivable hardware or software niche that can be tailored to the specific needs of the multimedia business. A good resource for information on this industry comes from Streaming Media, Inc., who bill themselves as the “home of the streaming media industry”. A few weeks of exploration on industry offerings could save you countless hours and dollars down the road.

If you plan to use Linux, look carefully at the vendors to determine their stance on open source, open standards, licensing and their qualifications to offer the depth of support necessary to guide you through the chaotic hawking that you are about to experience from these vendors. Be wary of “multimedia appliances” that prepackage everything you need. Unless the package contains a full solution for all aspects of this problem, you could be setting yourself up to be spending a lot of money so, to use an old army expression, your data can “hurry up and wait” for the next link to your client.

Resources



Frank LaMonica holds a BSEE from the University of Texas in Austin and has been working in the computer graphics industry for over 18 years. His previous experience includes work with most computer operating environments, including UNIX, Linux and Windows/NT. Frank was CEO of Precision Insight (PI), the company that developed the 3-D Direct Rendering Interface (DRI), which is now part of XFree86 4.0 and has been instrumental in bringing open-source accelerated 3-D graphics to Linux. Frank now serves as the strategic director of MultiMedia for VA Linux since PI was acquired by that company in April of 2000. Besides promoting open-source multimedia, Frank continues to pursue his moonlighting career as a concert classical guitarist.

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

[Advanced search](#)

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

The Story of OpenAL

Bernd Kreimeier

Issue #81, January 2001

Open Source and Open Standards: Chapters One through Three. A voyage through one of Loki's free software projects.

At Loki Entertainment Software, we deal with lots of different kinds of software. From public domain, to BSD-licensed source, to Free Software under GPL and LGPL, to proprietary code under contract and NDAs, to closed source hardware drivers, we encounter it all on a daily basis. Moreover, we also write a variety of source. The games we port to Linux are not open source, but our own work is available as free software in projects like part of projects like Fenris, Setup, SMPEG or SDL.

Crossing the fences from all sides over and over again, and getting to know both the concerns of free software and open-source advocates as well as the concerns of publishers and developers, our priorities have changed. As important as open-source and free software are, one of our projects in particular developed out of the recognition that there is something even more important: open, well-documented standards. That project is OpenAL, and its history so far is a fair bit of education on how important standards are—and how difficult it is to create them.

A Case of Need

With the Linux ports of *Heretic2* and *Heavy Gear 2*, Loki encountered engines that used spatialized sound above and beyond left-right panning. *Heretic2* for a time served as a showcase for audio hardware manufacturers: Aureal's A3D and Creative's EAX, specifically, were both supported. Both competitors at that time had ventured beyond the DirectSound3D feature set and were following very different approaches to spatialized sound. Windows game developers, caught between three different APIs (not counting the large number of commercial audio SDKs offered for Windows), more often than not chose to ignore the issue altogether and refrained from implementing advanced

features (unless they decided to go with one of the SDK offerings that promised to provide a convenient abstraction). The manufacturers stepped up and in many cases guided the developer's hand in adding EAX or A3D support to their engines. Spatialized 3-D sound became a checkbox item, a feature to be listed on the game box, and for a while the Windows game developers were offered an easy ride.

Unfortunately, the situation for Linux users was quite different. Driver support for the SB16 de facto standard had stabilized, and PCI sound cards, all in all, worked reasonably well under Linux. Yet, no ALSA or OSS-proprietary or open-source support for the most advanced features and boards was readily available.

Developers targeting Apple users found themselves in a similar situation—while the multimedia APIs on Mac OS might be more sophisticated and complete than the Linux solutions, they were even less compatible with DirectSound and equally oblivious to EAX and A3D features. Windows developers who cared about portability faced an audio feature set full of shining promises, but without a portable API. Given the vastly different approaches of the two main competitors, things were bound to get worse—at a time when Microsoft didn't even provide DirectSound3D for the NT platform that many developers preferred.

Creative, a dominant force in the audio hardware market, found itself challenged by Aureal, a company that attempted to take its experience with high-end spatialized sound solutions into the PC consumer market. Aureal proposed “wavetracing”, a proprietary approach to generating spatialized sound by simulation of sound propagation and acoustic properties of a scene—essentially, by submitting scene geometry to the sound driver. It is important to understand that to a much larger degree than graphics, audio technology was, and still is, software-based. Both the viable market for SDKs, like Miles Sound System or QSound's QMixer, and the evolution of the competing hardware drivers are clear indications of this. Aureal's advantage was their knowledge of simulation, the calculation of early reflections and a lot of optimization work for their drivers. Their disadvantage was the PC consumer market, an environment in which cheap speakers are added to budget solutions as an afterthought. Moving high-end audio processing out of the laboratory and away from the individual Head-Related Transfer Functions (HRTFs), on a desktop where most users and most developers didn't really spend much thought on spatialization, proved to be extraordinarily difficult and, as we now know, ultimately fatal.

Despite operating at diminishing returns for all their efforts, Aureal managed to accomplish one thing: to awaken interest and tentative user demand for better 3-D audio in games. For early adopters, it became a selling point, and

publishers and developers followed eventually. Creative, which had their own road map to future audio features, countered by emphasizing their own extended feature set—a much simpler solution focused on stochastic reverberation to emulate the effect of late reflections. In a way, the early and late reflection solutions are complements but the API's and implementations were fundamentally different, and Creative had picked well: EAX was easier and simpler to use, and for all the lack for simulation accuracy, stochastic reverb proved to be a much more effective audio cue for the noisy, low-end speaker desktop.

Both companies were considering Linux initiatives. On one hand, Aureal had a complete software implementation of “wavetracing”, eventually marketed as “A2D” to support the A3D feature set on their competitors' sound cards that could have been the base of a proprietary or even open-source sample implementation. On the other hand, Aureal's code base was a heavily optimized assembly, and a Linux port was not a foregone conclusion. Creative, on the other hand, recruited Linux coders to develop drivers in-house and looked for ways to support EAX under Linux. Both APIs were written with respect to Microsoft's DirectX and COM, but Aureal's geometry-based A3D API had also been inspired by the OpenGL API. This was one of many examples on the influence of OpenGL on the development of OpenAL.

Enter the game developers.

Scratching the Itch

Even before the days of EAX and A3D, some game developers were looking for a portable sound solution. The OpenGL-GameDev mailing list, which hosts more than its average share of developers interested in portable code, spawned in 1998 a list dedicated to discussion of a new, open audio library—tentatively named OpenAL. Proposals were written and posted, and several developers spent significant time coming up with a good solution. However, it quickly became apparent that between us, “good” was measured by very different metrics. Some of the subscribers were mainly interested in music and composition, solidly grounded in a sound synthesis background. Others cared only about spatialized sound, as it was in the days of DirectSound3D. Some preferred explicit, sophisticated OOP design in the API, while others wanted solutions that were less assuming. For all those efforts, little more than a few headers and a white paper were written, and over the months the project lost focus and tapered out. Lack of consensus and maybe a lack of experience in cooperative efforts contributed to this early failure, but to a much larger degree it was the lack of precise objectives, and the sheer variety of “audio” features to deal with, that doomed the first OpenAL effort not once, but twice. In early 1999, a resurrection occurred, coinciding with a brief mention of OpenAL in

Jonathan Blow's roundup of Sound API's and SDK's for *Game Developer Magazine* (May 1999). Published as a sidebar, Terry Sikes (who had written the 1998 white paper) and I stated the objectives of OpenAL. At that time, Aureal was considering a revised, portable A3D API, and our statement emphasized interoperability with OpenGL as a desirable property for a portable audio library, especially if processing geometry. Neither Aureal nor the OpenAL mailing list made significant progress in their separate efforts, and it wasn't until late 1999 that another company decided to get involved: Loki.

Work on *Heretic2* and *Heavy Gear 2* was commencing at that time, and while we were not (and, truth to be told, still are not) in a position to support EAX or A3D, it became obvious that some solution was needed. The minimum we needed was a Linux implementation of the feature set of DirectSound3D: distance attenuation and Doppler Effect, spatialization in 3-D, pitch and directional sound cones. Beyond that, it would obviously be desirable to talk to IHVs like Creative and Aureal and nurture their tentative interest in Linux drivers. Loki approached contributors to the original OpenAL discussion, set up a mailing list, committed one developer, the esteemed Joseph I. Valenzuela, full-time to the software sample implementation, and the three of us set out to work. Michael Vance, the lead coder on *Heavy Gear 2*, and myself (working on *Heretic2*) revisited the original OpenAL discussion and the existing proposals. A number of decisions, in particular the commitment to OpenGL-like conventions and API design, were taken with an eye on the past failures. Of course, the problem domain "audio" is very different from graphics by any account, but a lot of discussions were avoided by applying GL-like syntax wherever applicable. Initially, we expected a good deal of the API to handle geometry in explicit ways, just as A3D did, but we also applied the same reasoning to other parts of the OpenAL interface. Mimicking GL only got us so far: OpenAL is essentially a scene graph API, with a lot of explicit objects. We deviated from GL on other accounts as well: OpenAL has less entry points and more tokens, which has advantages for changing and deprecating API mechanisms while preserving ABI backward compatibility. We adopted a separation into AL, ALC, ALU and ALUT libraries (in loose analogy to GL, GLX/WGL, GLU and GLUT) but almost exclusively focused on the core AL API. Short-term itches led to immediate scratching, and *Heavy Gear 2* was heading toward being the first Loki game to ship along with the first OpenAL sample implementation. Even at that early stage, our OpenAL maintainer found himself in valiant struggle to keep the library in sync with the ever-changing specification drafts.

It was well before the Game Developer's Conference (GDC) in March 2000 in San Jose, that Creative expressed interest in OpenAL. To complement their own Linux driver work, and with respect to portability and acceptance, a vendor-neutral, OS agnostic API had become increasingly attractive for IHV's. Microsoft did not exhibit interest in extending DirectSound3D, and the Interactive Audio

Special Interest Group (IASIG), which had published Interactive 3-D Level 1 and Level 2 guidelines, did not consider itself in the business of specifying API's. Loki had tried its best to keep the implementation and the specification suitable for purposes beyond our short-term needs, and we were quite serious about the "Open" part of the project, but the interest expressed by Creative and others was a surprise that changed the rules. Beyond helping us in getting Linux ports of games feature complete, OpenAL now had IHV backing that might eventually establish it as a standard.

A Larger Stage

Loki and Creative announced their initiative during GDC, and we published the initial specification draft written by Michael Vance. Since then, the scope and requirements have changed quite a bit, and a lot of additional work was done during the last half-year. While our original focus was solely on 3-D spatialized sound, we now have to support stereo and multichannel formats, some of which (like MP3) are quite proprietary. Compression and streaming proved to be difficult issues. More than one solution was implemented, and it was a suggestion from Ian Ollman on the OpenAL mailing list which led to the adoption of buffer queueing to handle streaming audio. The more actual and potential features we had to account for, the more paranoia had to be applied to the API. Backward compatibility was broken not once, not twice, but several times, with extensions added to keep deprecated features available for the games we had already shipped: *Soldier of Fortune* followed *Heavy Gear 2* and was followed by *SimCity 3000 Unlimited*. By now, *Unreal Tournament* uses OpenAL as well, and upcoming ports of games based on the UT license will inherit this feature. Cognitoy's *Mindrover* uses OpenAL under Linux, and they, as well as a few other Windows developers, await the release of Win32 OpenAL drivers with hardware support. At the moment, practically all games that use OpenAL do so under Linux, but other platforms will hopefully follow soon.

Between three of us at Loki, and Jean-Marc Jot, Garin Hiebert, Carlo Vogelsang and others from Creative Labs, the OpenAL specification has progressed to a final draft of the version 1.0—a reasonably complete, solid foundation that covers the DirectSound3D feature set, at the same time deviating from it in many details. For example, buffers are strictly separate from sources and are shared among them. Some of the differences are rather subtle, like the distinction between clamping and reference distances, or the explicit definition of the reference velocity used in Doppler calculations. In some cases, like the handling of multichannel data, we still strive to get away with a minimal extension to the API.

To a degree, the biggest accomplishment at this stage is not so much the Specification Draft itself, but what we have learned in the process. While we are

still far from finished, we have established a work flow and a reference for RFC's, proposals, discussions and revisions. We borrowed a little from the IETF and quite a lot from the way OpenGL is extended and modified. The toolchain we currently rely on—the SGML version of the DocBook DTD, and the respective Linux parser and formatting tools—comes with a couple of loose ends, and while the document has become more modular, the layout and rendering leaves a lot to be desired. Extensions have been proposed for various features, but implementations have not yet been provided. Our decision to follow the example SGI set with OpenGL—in several senses—has preserved the initial momentum that vanished from the earlier OpenAL efforts. In a way, choosing the name “OpenAL” has to be understood as a clear commitment to a vendor-neutral, open solution, and I think we have kept the promise.

The next milestone will be the official closure on the final draft of the 1.0 specification, followed by releases of hardware OpenAL drivers for Windows, Linux and other platforms. To fulfill its promise, OpenAL will depend on efficient driver support, a task that has to be addressed by Creative and other IHVs. Loki encourages IHVs to try open-source solutions, but the ultimate decision is theirs. This is a necessary side effect of an open standard—just as every free software project has the right to implement the OpenAL API, companies can choose to implement proprietary versions. Given the relative importance of optimized implementations and improved algorithms for audio SDKs and drivers, it will take time to make a case for shared infrastructure and open source.

Beyond the current specification, we have now a much clearer roadmap for the OpenAL 1.1 revision. The feature set defined by the IASIG I3DL2 guidelines, which are largely based on submissions by Creative, will be added as a vendor-specific extension, aiming for a vendor neutral solution in 1.1. A lot of feature requests and additions have to be reviewed, as issues we set aside for OpenAL 1.0 have to be resolved. Given the small amount of users at this time, we have already gotten some rather unexpected and quite surprising inquiries. The context handling API, ALC, will have to prove itself on several platforms before we can confidently expect that, unlike OpenGL, we will have a portable solution at the context/device level as well. The API, currently minimizing entry points at the expense of tokens, might find a different balance once the semantics are found stable. Meanwhile, we continue to port new games, and while each one has its own different idea about resource and state management, we find the existing API to be quite sufficient. Aside from new, orthogonal features like microphone capture, the majority of the work required is related to streaming and additional codecs. Ironically, the API road map does not aim for handling reflection and occlusion geometry at this point. While it still might make sense to explicitly support large reflectors, the game engines are usually much better

equipped to determine obstruction and occlusion, and our objective for OpenAL has shifted to allow the application coder to express the consequences of such conditions. The battlefield has shifted to questions about use of OpenAL for composition instead of straightforward spatialization, generalization of vendor-specific features or abstraction of proprietary solutions. HRTFs, initially considered to be outside the scope of AL and better left to device configuration and driver implementation, have emerged as a possible addition, based on Sensaura's work in this area. Support for Dolby, which was on Aureal's road map, would probably be desirable as well. The implications of the Khronos Initiative for streaming multimedia, spearheaded by SGI, with respect to OpenAL has to be clarified—like OpenGL, extensions for interoperability might be added to OpenAL in the future.

Beyond the design and coding work, OpenAL's organization will have to evolve. What is currently a loose cooperation between volunteers, with a few companies thrown into the mix, will have to be established as a nonprofit organization. Again borrowing from OpenGL's example, there will be an architecture review board, a voting process and all the other structures that come with incorporating open standards. Unlike OpenGL, the sample implementation and the eventually written conformance test suite will be open source. Much like OpenGL, the use of the trademark will be restricted to those who passed the conformance test in some authorized, official way.

The importance of all this is exemplified by Brian Paul's Mesa: like no other free software project I can think of, his clean room implementation of the OpenGL API has, over more than five years, led to the hardware accelerated, 3-D-graphics-capable Linux machines we have today. It was Mesa that provided the framework, and 3dfx's Linux Glide that provided the driver, which brought hardware rendering to Linux (*Quake*) gaming in the first place. Without SGI's diligent work on their OpenGL specification, and their support for Brian Paul, we might not have the DRI open-source solutions for Linux and probably would not have the proprietary NVidia drivers either. If anything, we want OpenAL to match what OpenGL accomplished for Linux and other non-Windows platforms. For Loki, the most important goal now is to get hardware support on Linux—and Windows: ultimately, we want our fellow game developers to choose open standards over closed ones.

Conclusion

Was it worth it? We did get more on our hands than we bargained for, but if the work we have done so far helps in establishing another portable API, the answer is a definite yes. Linux has only just started to face the issues of standardization and certification, and personally, my respect for the IETF has multiplied several times. The Linux Standard Base (LSB) definitely has their

work cut out for them and will need all support it can get. Beyond the LSB, the Linux Desktop will need a Multimedia API supplement and Special Interest Group to give ISVs and IHVs alike some ground to stand on—not to mention all those free software and spare time projects struggling to be portable. It might well be that Linux hackers and users alike underestimate the importance that the DirectX set of services had for Windows at large and games on Windows in particular. It is also possible that many Linux developers still underestimate the importance of the desktop. However, Linux is only one of many platforms that need a comprehensive set of portable multimedia APIs. If OpenAL succeeds in defining another building block for such an OS agnostic “OpenX”, we will have accomplished much more than we initially expected.

Resources



Bernd Kreimeier is a game developer, writer and physicist. As one of the coders at Loki Software he has worked on the Linux versions of *Heretic 2*, *Quake 3* and other titles. He is acting maintainer of the OpenAL specification draft.

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

[Advanced search](#)

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

[Advanced search](#)

A Crash Course in SDL

John Hall

Issue #81, January 2001

An adaptation of a chapter from the author's upcoming book entitled Programming Linux Games.

Linux gaming is exploding, partly because of the simple fact that geeks like games, and partly because of recent developments in Linux multimedia. Over the past few years a number of excellent Linux-oriented multimedia toolkits have emerged, such as the GGI graphics interface and the ALSA sound system. The SDL library has also made a bit of a splash recently. SDL is a general-purpose, multimedia-programming library that provides fast and portable access to graphics, sound, input devices, threads and OpenGL rendering. The core SDL library is portable to several flavors of UNIX as well as BeOS, MacOS and Win32. This makes it an excellent choice for developing cross-platform games without compromising performance.

Unlike many multimedia toolkits, SDL does not actually talk to the system's hardware. Instead, it serves as a layer between an application and the underlying system. For instance, SDL's graphics system might use the frame buffer console or X11 under Linux, but DirectDraw under Windows. In either case, SDL's API is unchanged, and the application need not worry about what's going on underneath, and in some cases a carefully written SDL application can be ported to a new platform with a quick recompile.

In this article we'll take a tour of SDL's video API from the ground up. We'll also demonstrate how to collect input from the keyboard. Most of this article has been excerpted from a chapter in the author's upcoming book, on Linux game development (No Starch Press and Loki Entertainment Software, scheduled for early 2001).

Getting SDL

SDL is free software (under the LGPL), and it's available for download at their web site (<http://www.libsdl.org/>). In addition to the actual SDL library, the SDL home page is full of example source code, demos, games and extensions. SDL is easy to install from source, but the SDL home page also provides binaries for several of the more common platforms.

SDL's Design Philosophy

If you've ever worked with Microsoft's DirectX toolkit, you'll notice that SDL is a tiny library in comparison. The source code to the core library weighs in at just under six megabytes, and that includes a lot of extra code that would never be linked into a Linux application. Don't be fooled, though—those six megabytes are well used, and the core SDL library provides almost everything you need to develop high-quality Linux games and media players. In addition, the web site is home to a number of add-on libraries that provide extra features such as image loading and advanced audio mixing. By keeping these features separate from the core library, SDL remains small and easy to learn.

The SDL library consists of several sub-API's, providing cross-platform support for video, audio, input handling, multithreading, OpenGL rendering contexts and other things that game programmers appreciate. Unfortunately, we don't have enough room to cover all of this, so we'll stick to video programming and input handling, the two things you really need in order to get your feet wet with SDL.

The SDL Video API

The SDL video API's sole purpose is to find a suitable video device and set it up for your application to use. After it has initialized the display (created a window or switched the video card into a particular mode) SDL gets out of your way, providing only a minimal set of functions for pushing blocks of pixels around. SDL is not a drawing toolkit; what you do with the video device after it is initialized is not SDL's business.

SDL uses structures called surfaces (of type **SDL_Surface**) to represent graphical data. A surface is simply a block of memory for storing a rectangular region of pixels (individual colored dots). Each surface has a width, a height and a specific pixel format (more on this later). SDL loads image files directly into surface structures, and the screen is also a surface (albeit a special one).

The most important property of surfaces is that they can be copied onto each other very quickly; that is, one surface's pixels can be transferred to an identically-sized rectangular area of another surface. This operation is called a

blit (block image transfer). Blits are a fundamental part of game programming because they allow complete images to be composed out of pre-drawn graphics (often created by artists with image processing software). Since the screen is a surface like any other, entire images can be sent to the screen with a single blitting operation. SDL provides a generic function for performing fast blits between surfaces, and it can even convert between surfaces of different pixel formats on the fly.

Setting up the Display

Before we can begin blitting surfaces to the screen, we need to initialize the SDL library and switch the display into an appropriate mode. Take a look at Listing 1, the equivalent of “Hello, world!” in SDL.

Listing 1. Setting Up the Display

This program includes the **SDL.h** header file, which is the master header for SDL. Every SDL application should include this file. The program also includes two standard headers, for the **printf** and **atexit** functions.

We begin by calling **SDL_Init** to initialize SDL. This function takes an ORed list of arguments to indicate which subsystems should be initialized; we are only interested in the video subsystem, so we pass **SDL_INIT_VIDEO** (if we wanted audio, for instance, we would call this function with **SDL_INIT_VIDEO | SDL_INIT_AUDIO**). Unless a fatal error occurs, this function should return zero. We also use C's **atexit** facility to request that **SDL_Quit** be called before the program exits. This function makes sure that SDL has a chance to shut down properly (which becomes especially important if a fullscreen application crashes).

Next, we use the **SDL_SetVideoMode** function to inform the display of our desired resolution (in this case 640 pixels across by 480 pixels down) and color depth (16-bit packed pixel). There is a catch here: SDL will try to set up the display as requested, but it might fail. If this happens, SDL won't tell us, but it will instead emulate the requested mode internally. This is usually acceptable, since the emulation code is relatively fast, and we would usually rather not deal with multiple modes internally. **SDL_SetVideoMode** returns a pointer to the surface that represents the display. If something goes wrong, this function returns NULL.

Finally, we report success and exit. The C library calls **SDL_Quit** automatically (since we registered it with **atexit**), and SDL returns the video display to its original mode. (We could also call **SDL_Quit** explicitly if we wanted to shut the system down before exiting our application; there's no harm in calling it more than once.)

Now that we've created an SDL application, we need to compile it. SDL applications are easy to build; assuming a proper installation of SDL, they just require a few flags and libraries. The standard SDL distribution includes a program called **sdl-config** (similar to the **gtk-config** and **glib-config** programs that ship with the GTK+ toolkit) for supplying the appropriate commandline arguments to gcc. The command **sdl-config --cflags** produces a list of the options that should be passed to the compiler, and **sdl-config --libs** produces a list of libraries that should be linked in. We can use backtick substitution to drop this into the gcc command line. If SDL is installed on your system, you can compile this example with the following command:

```
$ gcc sdltest.c -o sdltest `sdl-config --cflags --libs`
```

Drawing Pixels Directly

Putting data into an SDL surface is simple. Each **SDL_Surface** structure contains a `pixels` member. This is a void pointer to the raw graphic image, and we can write to it directly if we know which type of pixel the surface is set up for. We must call the **SDL_LockSurface** function before we access this data (because some surfaces reside in special memory areas and require special handling). When we are finished with the surface, we must call **SDL_UnlockSurface** to release it. The width and the height of the image are given by the **w** and **h** members of the structure, and the pixel format is specified by the `format` member (which is of type **SDL_PixelFormat**). SDL often emulates nonstandard screen resolutions with higher resolutions, and the `pitch` member of the pixel format structure indicates the actual width of the frame buffer. You should always use `pitch` instead of `w` for calculating offsets into the pixels buffer, or else your application might not work on some display devices.

The example shown in Listing 2 will use the SDL pixel format information to draw individual pixels on the screen. We have chosen to use a 16-bit (hicolor) mode for demonstration purposes, but other modes are equally simple to program.

Listing 2. Drawing Individual Pixels on the Screen

The code's comments give the play-by-play, but a few things might not be obvious. This program employs a very general routine for constructing hicolor pixel values; this routine will work with any hicolor (16-bit) pixel format that SDL recognizes. Although we could write a separate (faster) routine for each possible hicolor data layout, this would require a lot of work and would only marginally improve performance. The hicolor 565 (5 red bits, 6 green bits, and 5 blue bits) pixel format is perhaps the most widely used and could be reasonably optimized, but 556 and 555 are not uncommon. In addition, there is no guarantee that the bit fields will be in the red-green-blue order. Our

CreateHicolorPixel routine solves this problem by referring to the data in the **SDL_PixelFormat** structure. For instance, the routine uses the **Rloss** member of the structure to determine how many bits to drop from the 8-bit red component, and it then uses the **Rshift** member to determine where the red bits should be located within the 16-bit pixel value.

Another important issue involves the **SDL_UpdateRect** function. As we mentioned earlier, SDL sometimes emulates video modes if the video card is unable to provide a certain mode itself. If the video card does not support a requested 24-bit mode, for instance, SDL might select a 16-bit mode instead and return a fake frame buffer setup for 24-bit pixels. This would allow your program to continue normally, and SDL would handle the conversion from 24-bits to 16-bits on the fly (with a slight performance loss). The **SDL_UpdateRect** function informs SDL that a portion of the screen has been updated and that it should perform the appropriate conversions to display that area. If a program does not use this function, there is a chance that it will still work. It is better to be on the safe side and call this function whenever the frame buffer surface has been changed.

Finally, if you run the program you might notice that it runs in a window instead of taking over the entire screen. To change this, replace the zero in the **SDL_SetVideoMode** call with the constant **SDL_FULLSCREEN**. Be careful, though; fullscreen applications are harder to debug, and they tend to mess things up badly when they crash.

Drawing with Blits

We've seen how to draw pixels directly to a surface, and there's no reason one couldn't create an entire game with this alone. However, there is a much better way to draw large amounts of data to the screen. Our next example will load an entire surface from a file and draw it with a single SDL surface copying function. The code can be seen in Listing 3.

Listing 3. Drawing Large Amounts of Data to the Screen

As you can see, the bitmap file is loaded into memory with the **SDL_LoadBMP** function. This function returns a pointer to an **SDL_Surface** structure containing the image, or a **NULL** pointer if the image cannot be loaded. Once this file has been successfully loaded, the bitmap is represented as an ordinary SDL surface, and a program can draw it onto the screen or any other surface. Bitmaps use dynamically allocated memory, and they should be freed when they are no longer needed. The **SDL_FreeSurface** function frees the memory allocated to a bitmap.

The **SDL_BlitSurface** function performs a blit of one surface onto another, converting between pixel formats as necessary. This function takes four arguments: a source surface (the image to copy from), an **SDL_Rect** structure defining the rectangular region of the source surface to copy, a destination surface (the image to copy to), and another **SDL_Rect** structure indicating the coordinates on the destination that the image should be drawn to. These two rectangles must be of the same width and height (SDL does not currently perform stretching), but the **x** and **y** starting coordinates of the regions may be different.

Colorkeys and Transparency

Games often need to simulate transparency. For instance, suppose that we have a bitmap of a game character against a solid background, and we want to draw the character in a game level. It would look silly to draw the character as is; the background would be drawn too, and the character would be surrounded by a block of solid color. It would be much better to draw only the pixels that are actually part of the character and not its solid background. We can do this with a colorkey blit. SDL provides support for this, and it even provides support for run-length colorkey acceleration (a nice trick for speeding up drawing). RLE acceleration provides an enormous performance boost for blitting colorkeyed images, but this is only practical for bitmaps that will not be modified during the course of the program (since modifying an RLE image necessitates unpacking and repacking the image).

A colorkey is a particular pixel value that a program declares to be transparent (in SDL, this is done with the **SDL_SetColorKey** function). Pixels that match an image's colorkey are not copied when the image is blitted. In our example of a game character, we could set the colorkey to the color of the solid background, and it would not be drawn. Colorkeys make it simple to combine rectangular images of non-rectangular objects.

In the next example we will use a colorkey blit to draw an image of Tux against another image [see Listing 4, available at ftp.linuxjournal.com/pub/lj/listings/issue81/]. Tux is stored against a solid blue background, and so we will use blue (RGB 0, 0, 255) as our colorkey. For comparison, we will also draw the same penguin image without a colorkey.



Figure 1. Tux.bmp

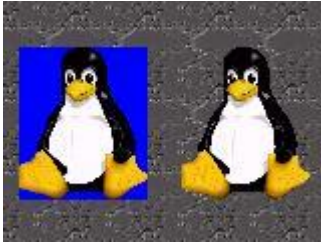


Figure 2. Colorkey-Output

Simple Keyboard Handling

SDL assigns a “virtual keysym” to each key on the keyboard. These codes (integers) map at some level to the operating system's keyboard scan codes (which in turn map to the codes produced by the keyboard's hardware), but SDL takes care of the mapping behind the scenes. SDL provides a preprocessor symbol for each virtual keysym; for instance, the Escape key corresponds to the symbol **SDLK_ESCAPE**. (You can find a list of valid keysyms in SDL's documentation.) We use these codes whenever we need to directly check the state (up or down) of a particular key, and SDL uses them when it reports key events. Virtual keysyms are represented by the **SDLKey** data type.

Since we won't be messing with the event interface for now (indeed, we haven't really even mentioned it), we'll need to ask the keyboard explicitly for its current state whenever we need to know about a key. A program can obtain a snapshot of the entire keyboard in the form of an array. The **SDL_GetKeyState** function returns a pointer to SDL's internal keyboard state array, which is indexed with the **SDLK_ keysym** constants. You only need to call this function once; the pointer remains valid for the duration of the program. Each entry in the array is a simple **Uint8** flag indicating whether that key is currently down. You should call **SDL_PumpEvents** periodically to update the data in the array.

More, More, More!

That should be enough to get you started with SDL. We've skipped over a lot, notably animation, alpha blending and audio playback. If you want to learn more about programming with this library, the best place to start is the SDL Documentation Project at <http://www.libsdl.org/>. You might also want to stop by the #sdl channel on irc.openprojects.net, where you're likely to find a good number of SDL fans with varying amounts of experience. Have fun, and happy hacking!



John Hall is a computer science major at Georgia Tech with an interest in Linux gaming. When he's not in a trance in front of a keyboard, he can often be caught blading around campus or caring for his pet arachnids. John can be contacted at overcode@lokigames.com.

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

[Advanced search](#)

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

[Advanced search](#)

Ogg Vorbis—Open, Free Audio—Set Your Media Free

Jack Moffitt

Issue #81, January 2001

Ogg Vorbis is the Open-Source Community's hot alternative to MP3.

Audio has become one of the killer apps of the network. With the distribution power that the global network offers, the music industry is being reshaped forever.

The boom of audio applications and files on the Internet is responsible for much current litigation surrounding copyright law and music licensing. The record industry is just now figuring out what most early users knew the first time they played an audio file on their computer: it's a new world for artists, listeners and record labels.

At the center of the upheaval are the technologies that make it all possible and a new technology, Ogg Vorbis, is ready to put this revolution into an even higher gear.

Ogg Vorbis is an open-source and patent-free audio codec that is being developed by Xiphophorus along with several other multimedia projects (cdparanoia and Icecast, to name two). Xiphophorus is a collection of open-source, multimedia-related projects and programmers who are working to ensure that Internet multimedia standards reside in the public domain where they belong. The work on Ogg Vorbis is currently funded by iCAST, the entertainment arm of CMGI.

Ogg Vorbis is an open standard, and this is important for a number of reasons. There are few truly open standards in the realm of digital audio. Look at Windows Media, Quicktime or RealAudio. These standards are all closed and proprietary, and because of this, none of the standards interoperate well (or at all outside of their corporate walls) with one another. When was the last time that you could play Quicktime 4 in RealPlayer or vice versa? When will Linux have Quicktime or Windows Media support? Linux and the Internet are

founded on open standards, and as multimedia on the Internet and on Linux rapidly matures, the need for multimedia applications like Ogg Vorbis grows rapidly as well.

There are two parts to Ogg Vorbis: Ogg and Vorbis. Ogg is a wrapper format, similar in some ways to Apple's Quicktime or Microsoft's Active Streaming Format. It helps you collect a group of things that belong together. For example, if you have an Ogg movie file, it might contain a Vorbis stream alongside a video stream in another codec. Or the Ogg movie file might contain ten Vorbis streams, one for each language available.

Vorbis is a codec that is written inside the Ogg framework. It is a general-purpose audio codec that is suitable for compressing most audio sources with good results. It doesn't use subbanding like some codecs do, but it does use vector quantization similar to others.

Vorbis is the only codec we've written so far, but not the only one we plan to write. There also are Squish and Tarkin.

Squish is a lossless audio codec, meaning that there is no loss in quality at all, and in fact, the decoded stream would be byte-for-byte identical with the original stream. You might want to use this for archiving master copies.

Tarkin is our fledgling video codec. It's a work in progress, but I can tell you it's based on wavelets, not on the MDCT like most modern codecs including MPEG-4 and JPEG. We're still playing around with it, but it's quite promising.

Codecs are hard to develop. They take a lot of math skills and a lot of time. Once you finish development, you still have to tune it, fix bugs and think of cool new things to add. This is why Ogg Vorbis focuses primarily on Vorbis and the Ogg framework at this point.

What's Wrong with MP3?

A lot of readers are probably wondering why we'd bother to develop Ogg Vorbis with MP3 already enjoying such widespread use. What's wrong with MP3? It's free, right? Wrong.

Have you ever noticed the amazing lack of free MP3 encoders, especially considering how popular MP3 has become? I can count them all on one hand. Some people will remember the famous letter from Fraunhofer back in late 1997. The letter asked for all the open-source and free MP3 encoders to cease and desist or start paying patent royalties. There are around 12 patents on the algorithms used by MP3, and all of them are heavily enforced by the owner Fraunhofer.

This patent enforcement has several negative effects. It's nearly impossible to have a free MP3 encoder because of the licensing fees for doing so. It costs \$2.50 per download (\$5 if you use the Fraunhofer code). Most of the free encoders disappeared without a way to pay this kind of tribute. MusicMatch, which makes a popular Windows encoder, sold a significant percentage of its company to Fraunhofer in exchange for an unlimited license.

Fraunhofer can change their rules at anytime, too. Prior to 1997, distributing MP3 encoders was fine. Right now, broadcasting in the MP3 format is free, but Fraunhofer stated that he intends to charge licensing fees for such use at the end of this year.

The deals the RIAA cuts for the broadcasting of commercial music are typically one-third to one-half a penny per song, which is quite reasonable considering that Fraunhofer may want to charge you 1% of revenue with a minimum of a full penny per song (these are my extrapolations from the current fees on commercial MP3 downloads). Is MP3 really worth three times more than the music it delivers?

It costs \$.50 a copy to license a decoder. These aren't the only costs associated with MP3, and really, some are just my speculations (hopefully the real fee for broadcasting will be considerably lower), but the patent holders can set or change the licensing fees to whatever they want, anytime they want. And, they already stated that they intend to do so at the end of this year for broadcasting. The point isn't whether it's \$15,000 or \$5. The point is they have the right to set the price however they see fit.

MP3 is an old technology. Audiophiles and programmers have been tuning encoders for a long time, but the technology is not improving anymore. Even LAME, one of the best MP3 encoders around, has new options that break the specification to try to squeeze more quality out. There just isn't anymore room in the format for new tweaks or improvements.

The alternatives aren't great either. Advanced Audio Coding (AAC), which is a part of MPEG-4, has quite a bit more IP restriction than MP3. There's more than one company involved in most of the technologies, which makes licensing even more cumbersome. The VQF format is locked up tightly by NTT and Yamaha. RealNetworks and Microsoft aren't known for their open standards either. Several derivative codecs like MP+ are problematic because they face the same patent restrictions that the regular MP3 codec has.

With all of these inherent problems and the need for a better way to work with audio on the Internet, it's not surprising that a solution would come from the Open-Source Community.

Ogg Vorbis vs. MP3

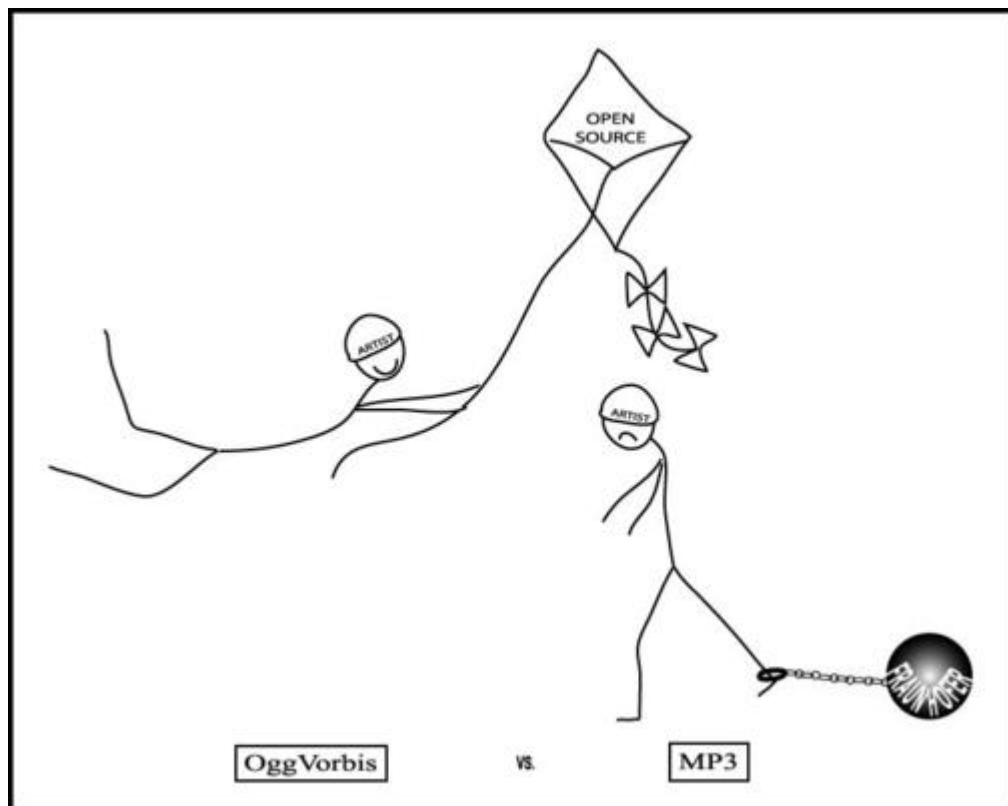


Figure 1. Ogg Vorbis vs. MP3

Ogg Vorbis is patent free and it was designed that way from the beginning. There are no licensing fees or costs associated with using the format for any purpose whether it is commercial or noncommercial. It's also open source under terms of the LGPL, so even the source code is free for companies and fellow hackers.

It's not enough just to be free. Vorbis has superior sound quality, which is what one would expect from a next-generation audio codec. Due to an extendable format, Vorbis' quality will improve for years to come without affecting decoders already being used. Vorbis sounds great now, but the quality is nothing compared to the Vorbis that will be around six months from now.

Quality is not the only advantage that Vorbis offers. Vorbis has some unique technical features as well: extensible comments, bitrate peeling and access to the raw codec packets.

Comments are defined in the format, so there are no worries about ugly and limiting hacks like ID3 tagging. The comments are stored in name=value pairs, and while there is a standard set of comments for applications to comply with for often-used data, you can add arbitrary comments if you need to.

Bitrate peeling allows for lowering the bitrate of a stream or file on the fly without re-encoding. This is achieved by encoding the most useful data toward the beginning of a packet. Slimming the stream is simply a matter of chopping the tails off of every packet before you send them out. Imagine listening to a radio stream that changes the bitrate based on your personal bandwidth needs. If you have dropouts, it sends you a smaller stream; if your download finishes, it sends you more data.

For multicast or other special applications, access to raw Vorbis packets allows complete control over how data is organized and shuffled around.

And, there's no reason to put up with leading or trailing silence since Vorbis has sample granularity on seek and decode. Remember all those gaps between tracks on your favorite trance CD? They disappear with Vorbis. Need to seek exactly to sample 303054? Vorbis provides a mechanism to do this. This makes Vorbis well suited to production work in ways that MP3 never was.

Developers and users, will appreciate having a high-quality set of reference libraries. This means that not everyone who wants to write an audio player needs to write their own decoder. Developers also have more time to spend on other things besides audio formats. This allows them to build more sophisticated and useful software.

Current Status

Two and a half years of Vorbis development (most of it as a side project) finally brought us the Ogg Vorbis beta1 release in mid-June of this year. It was limited to one bitrate, but it already had plug-ins for most players as well as support on many platforms.

In August, Ogg Vorbis beta2 release was launched at LinuxWorld Expo in San Jose, California. Five bitrates from 128kbps to 350KBps and several quality improvements were the main features.

Right now we're rapidly approaching the beta3 release, which has a number of significant quality improvements. This is mostly due to the many pairs of ears that report artifacts and bugs. The code has been organized toward the goal of a permanent API, and several new tools have been added.

Several optimizations were made that resulted in the decoder being twice as fast. We've also tuned the code to be tolerant for those who implement Vorbis using integer-only math. This allows hardware and embedded devices to more easily support Ogg Vorbis playback.

We've had over 100,000 downloads of Ogg Vorbis in the three months since its release, and third-party support has been wonderful so far. Xmms, Freeamp and Kmpg already support Vorbis playback (even popular Windows players like Sonique and Winamp support Vorbis). LAME can now produce Ogg Vorbis files as well as MP3 files and can re-encode MP3s to Vorbis in one step. Several people reported success with Grip the CD ripper, and new applications are popping up all the time.

A few content producers who are early adopters have started to embrace the format as well. Vorbisonic.com and eFolkmusic.com have Ogg Vorbis files up for download, and you can find more sites listed on the www.vorbis.com pages.

Shortly after our beta1 release, we did some random searches for domain names with "vorbis" in them that showed that a lot of people were buying Vorbis-related domain names. Several Vorbis-related sites have already turned up, including govorbis.com and vorbiszone.com.

Where We Are Going

We have only started the optimization process. On the decoding side, Ogg Vorbis is nearly as fast as the current MP3 decoders and should catch up soon. Several people already claim good playback on Pentium 120 machines. On the encoding side, real-time encoding is already possible on fast Pentium IIs and Pentium IIIs. Now that the API is getting stable and more features are getting knocked out, more and more people have started to turn to the issues of speed.

Comparing Vorbis to MP3 is almost unfair, since Vorbis has no channel coupling, but we're still ahead. There are some tricky patents that we must navigate, but the development team is looking to Ambisonics to fill this gap. Ambisonics was patented, but the patents have since expired. The company itself went out of business due to stiff competition from Dolby. Ambisonics technology would provide Vorbis with true three-dimensional, spherical sound, which can be mapped onto any number of speakers—all this in only four channels (one and two for stereo, three for surround and four for spherical sound). Taking advantage of channel coupling should easily drop bitrates by 40 percent.

Streaming is also very high on the list. We are currently testing streaming and should have a few test stations up before November. Soon after, Icecast should begin supporting Vorbis as its primary format for audio. This gives Internet radio fans higher quality streams, and it offers broadcasters a way out of end-of-year broadcasting royalties.

For streaming, lower bitrates are vital. Right now the lowest bitrate that the reference encoder outputs is approximately 128KBps. Typical streams range from 24KBps to 64KBps, and we'll soon focus on the tuning necessary to make low bitrate Vorbis sound fantastic. Lower sample rates are also on the horizon.

And, as always, we rigorously tune and improve the audio quality by adding quality-enhancing features and eliminating noticeable artifacts.

Ogg Vorbis 1.0, which includes the features outlined above, should be completed by the time you read this.

Gaining Ground on MP3

A lot of people ask us how we plan to take over the ground MP3 has already claimed. Some people don't even think that it's possible. I think it is. You can't really compare Vorbis to other audio codecs that have tried to accomplish what we have, because no other audio codec other than Vorbis is more free and more open than MP3. Part of the reason that the MP3 movement succeeded was due to the massive amounts of software that supported it. The software support happened because there was code lying around all over the Internet and documentation on how to use it or to write your own. Some people compare MP3 versus Vorbis to VHS versus Betamax. They say that just because we're technically superior doesn't that mean we will win. I guess those people don't realize that VHS won because the technology was actually more open.

Our strategy is to go after two groups: the artists and the developers.

Artists, and other content producers need, Vorbis to avoid paying percentages of their revenue to some technology company in Germany. Most of these people are also interested in having the best sounding quality product that they can get. People won't choose Vorbis or MP3 files simply for the sake of technology. People want music from artists they appreciate, or shows on topics they like, and they want the music to be available, transferable and easy to manipulate.

Developers want to include audio in their software—and not just for decoding and playback. Rich-media creation tools are only possible in the open-source world with open-media standards and patent-free algorithms like Ogg Vorbis. Including Vorbis into software is easy (it takes little time for a programmer to write a playback plug-in even if they are new to Vorbis and the Vorbis plug-in API).

If there is content being produced in Vorbis and applications all support Vorbis, the user probably won't even notice. Ease of use is achieved with transparency. Years from now, we might still be calling on-line music "MP3" just as some

people still call making photocopies “Xeroxing”, but the technology will come from different sources.

How You Can Help

Just like any open-source project, Vorbis reaches its full potential only with the help of the community. Programmers, audiophiles, musicians and evangelists are all needed. Encode some music with Vorbis, listen to Vorbis files and let us know if you hear anything that isn't in the original. Artifacts, once someone identifies them, are usually easily fixed. If you currently have a project that could (or does) play or encode audio, try Vorbis. Not only will the audience for Vorbis grow, but users will appreciate the functionality that Vorbis offers. Instead of creating music and putting it on-line in MP3, do it in Vorbis. By producing Vorbis files, you avoid limitations that patent holders enforce, and you increase user demand for Vorbis. Tell your friends, family and coworkers about Vorbis. Any effort to promote open standards like Vorbis for Internet audio is time well spent. And at this infant stage in Vorbis' life, we could really use the help.

Conclusion

Open standards for Internet multimedia are a worthwhile and attainable goal, especially with a high-quality open-source audio codec such as Vorbis.

Just as HTTP, FTP, TCP/IP and other open standards helped change the landscape for networking, our goal is to change the face of multimedia with tools that sound better, look better and work together better than the closed or patent-encumbered alternatives. You most likely use an operating system that relies on open standards and open source at its very core; why not expect the same from the multimedia applications you use?

Please visit the Ogg Vorbis demonstration site at www.vorbis.com or the Xiphophorus developers site at www.xiph.org.



Jack has been programming since he was six years old, writing everything from games to sound drivers, communications software and cryptography. In January of 2000, Jack became vice president of technology for iCAST, which is the entertainment arm for CMGI. At iCAST, Jack manages the open-source technology R&D team who is currently working on Icecast, Ogg Vorbis and a video codec codenamed “Tarkin”. Their goal, which was the same as the original

goal of Icecast, is to develop an open framework for multimedia on the Internet that ensures good quality, reliability and interoperability.

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

[Advanced search](#)

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

[Advanced search](#)

An Introduction to MSERV

Joshua Drake

Issue #81, January 2001

Drake explains how MSERV can end musical dictatorship.

Have you ever wondered who picks the music in your office? What about in elevators? I bet you sometimes stand in the elevator thinking, "You know, I bet if I just hit this little red button." That type of thinking is caused by elevator music. It never fails; unless you are employed by a progressive company, you will end up listening to artists that either: have been dead longer than you have been alive or make their money reproducing songs á la elevator music.

Obviously, this is a problem. A common solution for management is either to put everybody in cubicles, where they end up like trolls but can listen to their own music, or to dictate the music to be played.

The advent of personal jukeboxes like the NOMAD from Creative Labs (see Resources) can solve some of these problems. They have long play times, and the NOMAD can store over 150 CDs of music, something a walkman could never do. The problem with players is the word "personal". You can't share your general music interest; plus, they are \$500.00 US apiece.

The population of networks within the office and the proliferation of MP3 technology has spawned a new type of player. It is called MSERV. MSERV is a piece of open-source software that allows an office to turn a PC-running Linux, or another open-source operating system, into an MP3 jukebox; it's a lot like a personal jukebox but with centralized capabilities for the entire office.

When I first downloaded MSERV, I thought it was something other than what it is; I thought it was a broadcast server. In my mind it would be great to have over 500 CDs worth of music sitting on one hard drive that I or anyone else could access from their local web browser. We could set up our own play lists and listen to any music we wanted, separately. After realizing exactly what MSERV was, I had to readjust my thinking. I am glad that I did; MSERV provided

the people in our office a compromise with regard to the music we listen to. A technical description of MSERV would be: client/server, TCP/IP-based, central music deployment and ranking software. In short, it is like a Web Board Poll for MP3 listeners.

From an end-user standpoint, MSERV is a dream come true. It is specifically designed to play only the songs that people in the office want to hear. For example, if you have ten people who all enjoy the Eagles but only one who enjoys Yanni, that one person is out of luck. Why? Because MSERV uses a ranking system to pick the songs it plays. The ranking system is completely controlled by the user.

Let's say you are plugging away at your current geek-of-the-week task, and for some insane reason, a strange noise starts emitting from the overhead speaker. You're not sure, but it sounds like something your mother used to listen to. Instead of complaining and grumbling through your day, you fire up the good old web browser (hopefully not IE) and point it to the local MSERV machine. *Voilà*, you were correct, it is something your mother used to listen to. It's the Righteous Brothers and you saw *Dirty Dancing* one too many times to ever want to hear them again. Don't despair! MSERV allows you, the end user, to rank them into oblivion with one little click.

Of course, if everyone else loves them, they can rank them right back up, and you will hear it hour after hour, but at least some power has been given back to the end user.

The installation of MSERV is not difficult. The web site offers source, tar and rpm-based versions of the software. If you are running an RPM-based system (and the majority of systems out there are), then you'd type, as root, **rpm -i mserv-0.33-1.i386.rpm**. This will install all of the required components.

Once the program is installed, log in as a normal user. As the non-root user, you may start MSERV by entering the **/usr/bin** directory and typing **MSERV**. When MSERV starts, it will create a directory in the non-root users' home called **.mserv**. The **.mserv** directory is where the configuration and password files live.

MSERV comes with a web client. The web client is very simple to install. It only requires that Perl be on your system and that the ExecCGI option be turned on in the directory that you install into. Just copy the web client files to the URL that you would like to serve them from. For example, if your document root for your web server is **/usr/local/apache/htdocs**, you can install the files in **/usr/local/apache/htdocs/mserv**. Once you have edited the **httpd.conf** for the ExecCGI option, restart the Apache server. You should be able to point your web browser to your URL and have MSERV come up.

Once Apache is loading the pages correctly, you will want to change a couple of parameters in the **mserv.cgi** program. You can find it under the installed path. The main parameter you are looking for is the \$host line. The \$host line is the line that tells MSERV where to look for the programs.

If a web client is not what you are looking for, MSERV has a CLI and MS Windows client as well. The CLI is a Command Line Interface client and basically acts like a Telnet client. The MS Windows client is written in Delphi. Unfortunately, we were unable to test the Delphi version, as we don't run Windows.

The Delphi-based client is good for offices. As we all know, the majority of offices out there still run MS Windows, and having a Delphi client makes sense. Besides, consider it a proliferation option. First an MP3 server on Linux, then a mail server, then a web server, then a desktop, etc. Before long everybody will be running KDE2, and nobody will know the difference.

If you are a developer, MSERV offers a wealth of opportunities for improvement. That is not to say that MSERV is a bad product, exactly the opposite. But like most open-source projects, it needs polish. It needs a simple install and configure program, something beyond RPM or compiling. It needs a prettier front end, and it needs to be centralized, but also offer broadcasting capabilities.

MSERV offers an opportunity to provide a new product to the market. Here are some examples: a jukebox in the back of a car that directly interfaces with the stereo—just like CD-based jukeboxes but not limited to ten CDs per cartridge. Instead, you are limited only to the size of the hard drive you put in it. You can purchase an 82 gig hard drive for \$300.00 US—82 gigs of hard drive space is large enough to hold approximately 2,000 CDs worth of music.

Another example would be a set-top box. You could connect a little box to your stereo at home and have all of your CDs available instantly. You could even set it up so that as soon as you put in a new CD it would check the CD information from Cddb and start ripping the CD into an MP3 archive.

If you were to add remote control services to either of these, you would have instant emperor-like status among couch potatoes.

MSERV is only in release 0.33, and it is not considered stable. We at Command Prompt, Inc. World Headquarters, a three-person company, have been using it for almost three weeks. The consensus is that we like it and that it is stable. If you have a resident geek in-house, give MSERV a try. It's a fun little product and actually provides a useful solution to a common problem. The true power of

the program is in the open-source nature of the product. Anybody can develop for it, and anybody can extend it. But then again, you're reading *Linux Journal*. You already know this.

Resources

Joshua Drake is an e-commerce and Linux consultant who owns his own company, Command Prompt (<http://www.commandprompt.com/>). He has been using Linux for almost nine years and is the Linux Documentation Project's webmaster. His other projects include the LinuxPorts.com website and the OpenDocs publishing company.

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

[Advanced search](#)

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

Code Reviews

Larry Colen

Issue #81, January 2001

As funny as it sounds, the best code reviews are the ones that actually get done.

One of the most touted advantages of open-source and free software is peer review. "Many eyes make all bugs shallow" is the standard refrain. This assumes that you actually have many eyes looking at all portions of the code and that those eyes know what to look for. Code reviews are like sex, just about anyone can do it, but skill and training can make you a lot better at it.

For both of the programmers who have not already heard the lecture, here are the advantages of doing code and design reviews:

- The earlier bugs are found in the life cycle of a product, the cheaper and easier they are to fix.
- If someone else looks at your code or design, they are likely to find mistakes you missed.
- When you know that someone else is going to be looking at your code, you are a lot more likely to tidy it up and make sure there is accurate and up-to-date documentation.
- You can learn a lot by reading other people's code.
- More than one person who is familiar with a program is the best insurance against "Mack Truck Syndrome", which is when the only person who understands the software gets hit by a truck, leaves the company or is for some reason not available for consultation.
- It can be a means of establishing quality metrics, so one can measure the effectiveness of different quality processes.
- The process of explaining your software to someone else can help you actually review your own program, rather than just looking at and seeing what you expect or want to see.

- When done right, code and design reviews can save time and improve quality over the entire project life cycle.

The disadvantage of code reviews is that they take time, not only of the person actively working on the project, but also for other people who are usually under deadline pressure themselves. Despite the fact that there are numerous studies showing that the overall number of human-hours spent on a project is lower when reviews are properly done, there is the constant temptation to bet that there really aren't any problems. That means waiting until the code is written and being debugged to try to find problems.

I'm nearly done preaching about why peer reviews are a good thing. I will say that the most important thing about reviews is that they actually get done. A quick and dirty review that finds only a third of the bugs is more effective than a thorough, exhaustive review that no one actually performs.

There are two basic types of reviews: walk-throughs and formal inspections. In a walk-through, the author takes one or more colleagues on a tour of the document under review. In my experience, about 80% of the errors found in a walk-through are actually found by the author in the process of explaining the document. Eli Weber, a former colleague, said "If I could talk to the wall as if it were a person, I wouldn't need someone else in order to do a code review."

In a formal inspection, one or more people are given a document design or program to review. Oftentimes each person concentrates on different things: adherence to style or programming standards, logical errors, completeness of documentation, etc. It is common practice to make checklists of what each person is to look for, such as coding style rules, common mistakes, potential security holes, etc.

There is no problem with someone finding an error that they weren't concentrating on. For example, if the person checking for compliance to coding standards notices an AND operator being used in place of an OR, they should definitely note it. But not every person is trying to make sure all aspects of the document are correct. When a reviewer finds a problem, they note its location and its severity. Priorities range from "something you might consider thinking about changing if you find yourself with absolutely nothing to do and the boss is in a bad mood", to "if this error is not fixed immediately it will set in motion a chain of events leading to the end of civilization as we know it".

Once everyone has reviewed the document, a meeting is held to discuss the errors found. The errors can be discussed in just about any logical order: by page of the document, by severity of errors or each reviewer can list all of the issues they found. This meeting is usually run by the author of the document, but it could be run by anyone. As each issue is mentioned, the people at the

meeting come to a consensus as to the severity of the issue. Often, the reviewer may not be able to assess the severity of an issue but may just have a hunch that something is not as it should be and will ask the other attendees their opinions. It is crucial to remember that the point of the meeting is not to solve the problems but merely to log them. I know full well how hard it can be to keep a room full of engineers from immediately trying to solve an interesting problem, but the facilitator must remember to keep the meeting on track, or it will go on for weeks and will cost more than it saves.

This, at least theoretically, is how a code review works in an industrial or educational setting, where there are several people in close proximity

But what about the open-source world, where the people are working on a project are not likely to be in the same country, much less city? There are some tightly managed projects with mailing lists where someone will propose a change and anybody on the mailing list can take a look at it at their leisure. If they notice something they think may be a problem ("You infidel, you indented by three columns when by everything right and holy you should indent by two!") they can send e-mail to either the list or the author for calm and rational discussion. Other projects will follow the model where someone needs a program to do something and not finding it already available, they write it themselves. Once it is working to their own satisfaction, they make it available to the public, possibly posting it to freshmeat or another similar forum. Sometime later, someone else needs a program to do the same, or similar task. This time they find the one that was just written, they download it and it doesn't run. They figure out where it is crashing, find the offending code, and motivated by the kindness of their hearts and goodwill toward their fellow geek, inform the original author of the bug and send a suggested patch.

Notice that in neither of these scenarios is there any guarantee that if you write software it will be reviewed by someone, or if it is, it will be reviewed soon enough to be useful. How can a conscientious programmer make sure that their code is not only reviewed, but reviewed in time to be useful?

Most geeks have geek friends. Writers often form groups that meet once a week to review and critique each other's stories. The same could be done for software. Participants could e-mail the design or code to be reviewed sometime before the meeting. Using the formal inspection methodology, they could use the meeting time to collect notes about issues found. Alternatively, when a design or program is ready for review, an e-mail can be sent to a mailing list inviting geeks to meet at Hans' Pizzhaus for pizza, beer and a code walk-through.

If there isn't anyone locally to help, most of the work of a formal inspection can be accomplished via e-mail. The key is to get a group of people who will actually review each other's code rather than just quickly glance over it. Likewise, most of the errors found during a walk-through are found by the author while explaining the code. If there isn't anyone around you can convince to sit down and listen while you explain your software, and if your cat does not possess the raw programming talent of Richard Stallman and Dennis Ritchie combined, you can perform a walk-through with an "internet audience" by thoroughly documenting your software. The process of sitting down with your program and writing a document to explain what it is doing should force you to look at it from a perspective that allows you to find annoying bugs like a misplaced parenthesis.

Doing a code review can significantly improve the security of a product. First of all, if other people look at a program it's a lot harder to leave back doors in it, such as "if someone logs in with the username of f00Bidoo, automatically grant root privileges". However, most exploits, especially of open-source software, are written by probing or searching for known classes of errors—sort of a "malicious code review". In order to close such holes, it is necessary to find them before someone with unfriendly intentions does.

Searching for bugs in the code is, as mathematicians would say, necessary, but not sufficient, for security. Open BSD addresses security through aggressive code reviews. The security team at Open BSD also does tons of research. Theo de Raadt, says "In fact, it is what we focus on. Trying to find NEW kinds of programmer errors, in old code. Code that exists, which has common mistakes that people never think of. Like fd_set overflows for instance. In Open BSD, buffer overflows have basically been extinct for about three years." For more information about the Open BSD philosophy and goals about security, check out www.openbsd.org/security.html.

In summary, most of the errors found in a code review are found because someone actually sees the code, whether it is the peer reviewer or the author in the process of explaining it to someone else. Likewise, it is better to have a modest inspection methodology that actually gets implemented versus a grandiose scheme for which there is never quite enough time to get implemented at all. When one is concerned about security, there are tremendous gains to be had by carefully checking the code for errors, but that is not sufficient. One must also look at the code with the eye of someone that wants to compromise the security of the system. One must not only look for cases where the code won't do what it was supposed to, but where it can be made to do things that it wasn't supposed to as well.

I'd like to thank Theo de Raadt for his invaluable assistance in discussing Open BSD and reviewing code for security issues.



Larry Colen (lrc@red4est.com) has been playing with Linux since March of 1994 and working with it professionally since November 1998. His professional interests include Operating Systems, multiprocessing, software quality, computer security and signal processing. For fun he teaches performance driving, swing dances and rides bicycles. More information than you really want to know about him can be found on his vanity page at: www.red4est.com/lrc including a rambling missive on software quality at: www.red4est.com/lrc/prof_html/debuggable.html.

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

[Advanced search](#)

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

Breaking through the Maximum Process Number

Zhang Yong

Issue #81, January 2001

Breaking through the maximum process number restriction in i386-based Linux.

Process management is the most important part of an operating system. Its design and implementation can greatly affect performance. In a multiprocess OS, many processes run simultaneously, thus increasing the CPU usage and system performance. By running processes concurrently, we provide multiple services and serve more clients at the same time, which is the main task of a modern OS.

In the Linux Intel i386 architecture, multiprocess is already supported. By choosing proper process scheduling algorithms, it has lower average response time and relatively high system performance. But unfortunately, there is a limitation in the Linux kernel 2.2.x that limits the number of running processes to 4090. This number may be enough for a desktop system but is inadequate for an enterprise server.

Consider the basic principle of a typical web server, which is based on multiprocess/multithread technology. When a client request comes, the web server creates a child process or thread to handle the request. So it is easy for a heavy load server to have thousands of processes running. In fact, most of such enterprise servers run operating systems like Solaris, AIX, HP-UX, etc., rather than Linux.

Many Linux developers have noticed this problem and have tried to solve it. In experimental version 2.3.x and prerelease 2.4, this limitation has been dealt with. But, it will still be a while before the official release of 2.4, and it may take even longer for it to be stable. Does this mean we have to choose another OS? Is it possible to find a solution that can break through that limitation for Linux 2.2.x? In order to answer this question, first we have to know how process management in 2.2.x works.

Intel i386 Architecture and Linux 2.2.x Memory Management

Process management is tightly bound with memory management. Since the implementation of memory management is based on hardware architecture, we have to have a look at the i386 architecture first. In modern operating systems, virtual memory technology is widely employed. Thanks to virtual memory technology, software can use more memory than is physically present. That is to say, the memory addresses used by software are virtual and are converted to real address by processor-provided mechanisms during access.

There are two basic memory management methods: segmentation and paging. Segmentation means dividing memory into several segments and accessing memory by both segment pointer and offset. This method is used in early systems like PDP-11, etc. Paging means dividing memory into several fixed-size pages and using pages as the basic memory management unit. When accessing memory, an address is converted to a physical address according to the page table.

Memory management in the i386 architecture is called segmentation with paging. The virtual address space is divided into segments first by using two tables: the Global Descriptor Table (GDT) and Local Descriptor Table (LDT). After this, the virtual address is converted to a linear address. Then the linear address is converted to a physical address using two-level page tables: the Page Directory Table and Page Table. Figure 1 shows how the virtual address has been converted to a real address.



Figure 1. Virtual Address Convert

...
...
APM BIOS Data
APM BIOS 16-Bit Code
APM BIOS Code
APM BIOS Reserve
Reserve
Reserve
User Data
User Code
Kernel Data
Kernel Code
Unused Descriptor
NULL Descriptor

Figure 2. Global Descriptor Table

In Linux, the kernel runs in ring 0. By setting GDT, the kernel puts its code and data into a separate address space. All other programs run in ring 3 with their data and code in the same address space. Creating different page tables protects those user programs. The GDT table in Linux 2.2.x is shown below in Figure 2. In practice, a user program can use other code/data segments by setting LDT.

Kernel 2.2.x Process Management

A process is a running program with all resources allocated. It is a dynamic concept. In the i386 architecture, "task" is an alternative name for process. For convenience, here we will use process only. Process management is a concept concerned with system initialization, process creation and destruction, scheduling, interprocess communication, etc. In Linux, process is actually a group of data structures including the context of process, scheduling data, semaphores, process queue, process id, time, signals, etc. This group of data is called Process Control Block or PCB. In implementation, PCB is in the bottom of the process stack.

Process management in Linux relies greatly on the hardware architecture. We have just discussed the basis of page-with-segment memory management in i386, but in fact, segment plays a more important role than just a block of memory. For example, Task Status Segment is one of the most important segments in i386. It contains much data that is required by the system. Each process must have a TSS pointed by TR register. According to the definition of i386, the selector in TR must select a descriptor in GDT. Additionally, the selector in LDTR, which defines a process LDT, must have a corresponding entry in GDT as well.

In order to satisfy the above requirements, Linux 2.2.x GDT is allocated for all possible processes. The maximum concurrent process number is defined when booting the kernel. The kernel reserves 2 GDT entries for each process.

System Initialization

In Linux 2.2.x, some process-management-concerned data structures are initialized when booting the system. The most important of these are GDT and the process list.

When the kernel starts, it must decide the size of GDT. Since two GDT entries must be kept in GDT for each process, the size of GDT is defined by the maximum concurrent process number. In Linux, this number is defined as NR_TASKS at compile time. According to Figure 2, the size of GDT is $10+2(\text{with APM})+NR_TASKS*2$.

The process list is actually an array of PCB pointers, defined below:

```
struct task_struct *task[NR_TASKS] = {&init_task,};
```

In the above line, **init_task** is the PCB of the root process. After inserting this process into the process list, the process management mechanism can begin its work. Note the size of process list is also dependant on **NR_TASKS**.

Process Creation

In Linux 2.2.x process is created by a system call, fork. The new process is the child process of the original process. Using a clone can create a thread, which is actually a lightweight process. In fact, there is no real thread in Linux 2.2.x. Figure 3 shows how the fork system call works.

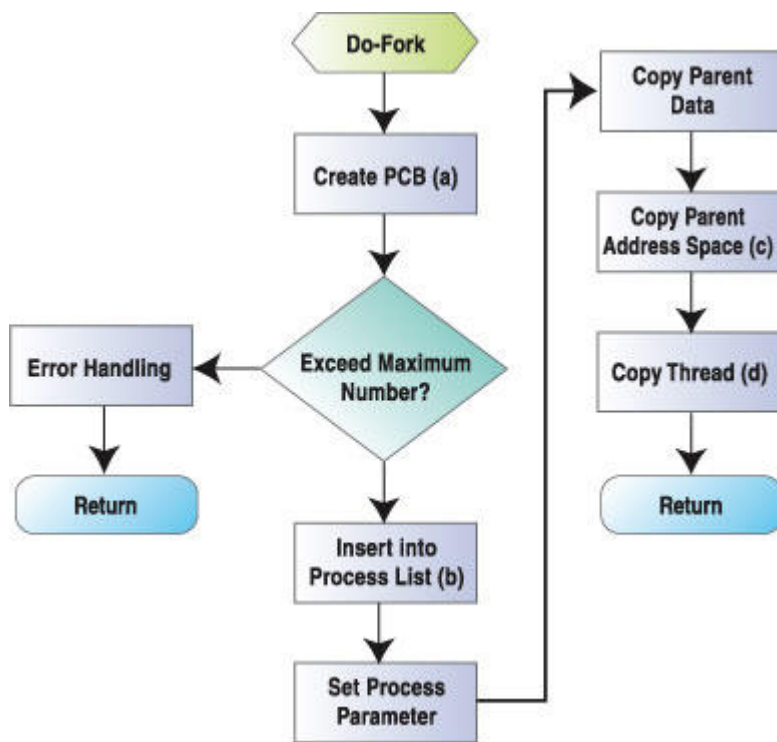


Figure 3: Fork System Call

The key steps in fork are:

1. Create the new process PCB: the kernel allocates two pages for the new process stack, and the PCB is put at the bottom of it.
2. Insert the new process into the process list: the kernel must find a empty entry from the process list. If the system has reached the maximum concurrent process limit, no empty entry will be found and the system call fails.
3. Copy parent address space: the child process has its own address space, but first it shares address space with its parent using copy-on-write mechanism. The corresponding GDT descriptor of the new process LDT is also created in this step.
4. Setting TSS for new process: the TSS of the new process is created in PCB and the corresponding GDT descriptor is also created.

Scheduling

The core of scheduling is the algorithm listed below. But here we just take a look at process switching. In Linux 2.2.x process switch is done in the switch_to function. It works like this:

1. load new TSS by setting TR
2. save old FS and GS registers into old PCB
3. load LDT if needed by new process
4. load new page tables for new process

5. load new process FS and GS

Note that the value of TR and LDTR are coming from PCB.

Breakthrough the Maximum Process Number Restriction

What is the maximum process number restriction? According to the above discussion, we can easily find why there is a maximum process number restriction. The NR_TASKS defined in Linux 2.2.x statically defines the maximum concurrent process number at the compile time. NR_TASKS also defined the size of GDT at compile time. As defined in i386 architecture, the maximum size of GDT is 8192*8 bytes, which means it can contain 8,192 descriptors. In Linux 2.2.x, when booting the kernel, GDT is used as described below:

1. NULL descriptor (entry 0), reserved descriptor (entry 1,6,7)
2. Kernel code and data descriptors (entry 2,3) and user code and data descriptors (entry 4,5)
3. APM BIOS descriptors (entry 8-11)

In total, 12 entries are used. And since each process needs two GDT entries, in theory we can get $(8192 - 12)/2 = 4090$ processes running concurrently.

Solution to This Problem

Although GDT size is restricted by hardware, we still can find a solution for this problem. For one CPU, only one process can be running at a certain time. That is to say, it is not necessary to reserve GDT descriptors for all possible processes at all. When a process is about to run, we set its descriptors dynamically.

After analyzing the PCB structure, we can find the TSS and LDT (if any) in it. So when doing a process switch, we can find these two segments by PCB pointer, like this:

```
TSS: proc->tss  
LDT: proc->mm->segments
```

In fact, when doing a process switch, we can find the PCB pointer from the process list. Since both TSS and LDT can be found, keeping them in GDT at all times is unnecessary.

Our solution is to reserve only two GDT descriptors for each CPU, using common entries for all processes. For example, in a machine with two CPUs, four GDT entries are reserved. When process A will run on CPU1, GDT entry three and four will be set to the descriptor of TSS and LDT of process A. Old

values of these entries are discarded. Remaining GDT entries are used just the same as the original system.

Implementation Brief

The basis of our solution is to set the process TSS and LDT descriptor dynamically (see Listing 1).

Listing 1. System Initialization

Process Switch

In the original design, when doing the fork operation, the `tss.ldt` and `tss.tr` in PCB are used to save selectors in LDTR and TR. According to the original algorithm, the selector of the LDT of a process may exceed its 16-bit limit. So we use extra variable `tss.__ldth` with `tss.ldt` to save the selector. Since `tss.__ldth` is not used in Linux 2.2.x, our modification won't break the kernel. The saving of LDTR and TR now works like this:

```
((unsigned long *) & (p->tss.ldt)) =
(unsigned long)_LDT(nr);
if (*((unsigned long *) & (p->tss.ldt)) <
(unsignedlong)(8192<< 3)
    set_ldt_desc(nr,ldt, LDT_ENTRIES);
    // original code here else{
    //do nothing
    //let the process switch code handle LDT
    //and TSS
}
```

One of the benefits of this implementation is that we can easily discover if this process number is greater than 4088 by inspecting the value of `tss.ldt`. This is important for performance.

If a process number is greater than 4,088, it has no reserved descriptor in GDT and must use the shared GDT entries. We can find these entries by this code:

```
SHARED_TSS_ENTRY + smp_processor_id();
```

Listing 2 shows the code for dealing with the shared GDT entries.

Listing 2. Using Shared GOT Entries

After doing these, we have broken through the maximum process number restriction. We can even add an extra parameter in the lilo configuration file to set this number dynamically. The following line will set the maximum process number to 40,000, which is much greater than 4,090:

```
Append = "nrtasks=40000"
```

Conclusion

According to the above solution, we can set the upper limit of concurrent process number to 2G, in theory. But in practice, hardware and OS still limit this number. When creating a new process, the kernel will allocate memory for it, like this:

```
Process stack (2 pages) + page table (1 page)
+ page directory table (1 page) = 4 pages
```

So if the computer has 1G memory and uses five pages per process where the OS uses 20M of memory, the maximum process number can be:

```
(1G - 20M) / 20K = 51404 ~= 50,000
```

More practically, a process will use 30K memory at least, so the number now is:

```
50000 * (2/3) = 33,000
```

This number is still much greater than 4,090.



Zhang Yong (leon@xteamlinux.com.cn) is a senior software engineer of Xteam Software Co., Ltd. His work covers many aspects of Linux, including kernel development, Linux I18N&I10N and network applications, etc. He is currently focusing on the upcoming new release of XteamServer, which is a high-end server solution based on Linux. Xteam Software Co., Ltd. is also the vendor of XteamLinux and XteamLindows. They are both the most popular Linux Distributions in China. For more information, please visit <http://www.xteamlinux.com.cn/>.

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

[Advanced search](#)

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

[Advanced search](#)

Three-Tiered Design

Reuven M. Lerner

Issue #81, January 2001

Learn how to implement middleware into the mod_perl/Apache design mix.

Several months ago, we took a long look at Mason, a Web development environment that combines mod_perl, Apache and templates. One of the examples was that of a system for press releases, in which Mason components retrieve the latest press releases from a database. The style of programming demonstrated in this article could be characterized as “two-tier”, in which the Mason components speak directly with a database, using Perl's DBI mod_perl's Apache::DBI.

But as several people on the Mason e-mail list noted, this approach—in which the SQL statements are directly inside of Mason components—is often unwise. Modifying the database definitions, or even the brand of database server we use, forces us to change the components themselves. In addition, non-web programs must re-implement SQL calls within their own programs, rather than reusing a commonly maintained and tested library.

We can solve both of these problems by adding an extra layer of software, sometimes known as “middleware”, between the database and the Mason components. This increasingly popular architecture is known as a “three-tiered” approach, because it means that we must work with three different sets of software services: the database, a “middleware” abstraction layer, and the implementation/presentation layer.

This month and next, we will look at a simple web-based address book and appointment calendar that demonstrate this three-tier approach. Along the way, I hope that you'll learn the advantages and disadvantages of this approach and be able to weigh it alongside other approaches when creating a web site. Once we have looked into this general architecture, we will be well prepared to look into Java Server Pages with Jakarta-Tomcat, and application servers. We'll

examine the pitfalls involved with this approach, as well as how it can make development easier and more scalable in the long run.

The Database

The first tier, and perhaps the most significant, is the relational database. I will use PostgreSQL for this example, but it would not matter much if we used something like Oracle or MySQL.

Before we design our database, we must have a specification describing what we want to do. After all, the design of a database depends in no small part on how we intend to use it.

For the purposes of this article, we will keep the specification relatively short and ambiguous: I want to have an addressbook that I can view, search and update via the Web. In addition, I would like to be able to add, view and modify appointments using my web browser.

In order to accomplish this, we will need at least two tables, one containing people and the other containing appointments. Here is an initial stab at a definition of the people table (see Listing 1).

Listing 1. Defining the People Table

In other words, we will always store a person's first name, last name, country and e-mail address. Beyond that, we can store their address, city, state, postal code and some comments about them. This assumes that everyone we know has an e-mail address—an assumption that is increasingly true, but not necessarily a good attribute if you have friends and business contacts from outside of the computer industry.

Each of the entries in our people table will be uniquely identified by the `person_id` column, which is automatically incremented by PostgreSQL. In addition, we ensure that we only add each person once by checking for the uniqueness of their e-mail address. This allows us to have more than one friend named John Smith. It also means that we cannot store separate information about a couple with a shared e-mail address. There also is no good provision for handling people with multiple e-mail addresses.

Adding a new person into the people table is relatively easy:

```
INSERT INTO People
    (first_name, last_name, address1, address2, email,
     city, state, postal_code, country, comments)
VALUES
    ('Shai', 'Re'em', '10
Helmonit', 'Apt. 7',
     'shai@lerner.co.il',
```

```
'Modi\'in', NULL,  
71700, 'Israel', 'Six-year-old  
nephew')  
;
```

Because most of the columns are NULL, I can even get away with entering a bare minimum of columns:

```
INSERT INTO People  
  (first_name, last_name, email, country)  
VALUES  
  ('Hadar', 'Re\'em',  
  'hadar@lerner.co.il',  
  'Israel')  
;
```

Now we will create an appointments table in which we will store appointments with members of the people table:

```
CREATE TABLE Appointments (  
  person_id INT4 NOT NULL REFERENCES People,  
  start_time TIMESTAMP NOT NULL,  
  end_time TIMESTAMP NOT NULL,  
  notes TEXT NULL CHECK  
    (notes <> ''),  
  UNIQUE(start_time)  
);
```

Once I have the appointments table defined, I can add a new appointment by inserting a new row into appointments:

```
INSERT INTO Appointments  
  (person_id, start_time, end_time, notes)  
VALUES  
  (1, 'November 22, 2000 19:00',  
  'November 22, 2000 19:30', 'Read Dr. Seuss')  
;
```

But because person_id is defined so as to be a foreign key from people, we can only add an appointment if we are meeting with someone already in the people table. This might be adequate for our purposes, but a more sophisticated (and well-specified) system would presumably give us more flexibility. And of course, this database will not let me indicate that I am meeting with more than one person at a time.

Middleware

Now that we have created our initial database design, we will consider the design of our middleware layer, which insulates the web application from the database. If we ever decide to switch to another brand of database server, or even replace the database with flat ASCII files or DBM files, the object layer will remain the same.

In addition, non-web applications will be able to use this layer in order to access the database, making it possible, for instance, to write a set of routines that export our appointment calendar into XML, or to import it from another program.

This middle tier is often called the “business logic” of an application. The database makes it easy and safe for us to store and retrieve information, and the Mason components make it easy for us to create dynamic output for the end user. The middleware layer will try to force the database to do as much of the computation as possible, using built-in functions, views and stored procedures. But the actual logic that determines our application's functionality will reside in the middle tier.

Perl gives us at least two options when creating this layer. One possibility is to create a basic Perl module that provides subroutines and variables that can accomplish the tasks we need. Such a procedural interface is relatively easy to write and executes at the same speed as all other Perl subroutines.

But Perl also offers us the option of creating an object module. While Perl objects are slightly harder to write, and their methods execute more slowly than straight subroutines, they make it easier to conceptualize and write programs.

Before we can create our middleware layer, we have to answer some serious questions. What sorts of objects will we create? We could create a single database object that handles all of our queries for us, turning them into the appropriate SQL. But we will occasionally want to retrieve information about people without any regard for appointments, which means that we should have, if nothing else, a people object, as well as a separate appointments object. Because our database table definition forces us to associate each appointment with one person, we can only define our appointments object after the people object.

People.pm

Listing 2 contains a listing for People.pm, an object module that performs some basic tasks for the people table that we created earlier. The object is not complete and has some rough edges, but should suffice for demonstrating how to access a relational database via an object middleware layer.

Listing 2. People.pm, a Perl Object Module That Communicates with the Package People

The basic idea is that you create a new instance of people, and then manipulate the people in your appointment book with that object. To retrieve all of the names of people in the database, you can use the `get_all_full_names` method, as in this code fragment (also see Listing 3):

```
use People;
# Create a People object
my $people = new People;
```

```

# Retrieve all of the full names
my @names = $people->get_all_full_names();
# Print the names
foreach my $name (@names)
{
    print "name\n";
}

```

Listing 3. Retrieve-people.pl, a Program That Uses People.pm to Retrieve Information from the Database

To set or retrieve information about a particular person, you must first identify which person you're talking about. Because our middleware layer is meant to shield the user from having to think or worry about primary keys and other database-specific IDs, we will allow them to set the “current person” via either the first and last name, or via the e-mail address.

The e-mail address is guaranteed to be unique in the database layer, and thus using `set_current_person_by_email` is the most reliable method available. Nevertheless, it's often useful to identify people by first and last name, so we also offer the `set_current_person_by_name` method. In the current implementation, using the name to set the current person will match the first returned row from the database, which might not necessarily be what you want.

Once a program has set the current person, it can retrieve information about that person using the `get_current_info` method:

```

# Set the current person by name
$people->set_current_person_by_name("Shai", "Re'em")
|| die "Error retrieving person.";
# Print the information
foreach my $key (sort keys %{$info})
{
    if (defined $info->{$key})
    {
        print "$key => $info->{$key}\n";
    }
}

```

Each instance of `people` will keep only two pieces of state: the ID of the currently selected person (`$self->{current_person}`) and the database handle (`$dbh`) that connects us to the database (`$self->{dbh}`). We keep the database handle around because connecting to a database is a relatively expensive operation. We can thus save ourselves some time by connecting to the database in the constructor, and then using that connection each time we invoke a method on this object.

Of course, this means that the database connection will have to be destroyed when the Perl object goes away—a somewhat tricky task, given that Perl does not have explicit destructors, since it is a garbage-collected language. The solution is to create a method called `DESTROY`, which is invoked by Perl

whenever the object is destroyed. Our DESTROY method merely closes our connection to the database, allowing the object to be removed without potentially causing a memory leak in either the database client or server:

```
sub DESTROY
{
    # Get myself
    my $self = shift;
    # Get the database handle
    my $dbh = $self->{dbh};
    # Close the database handle
    $dbh->disconnect;
    return;
}
```

We can even create a new person, invoking the new_person method with a set of hash keys and values as arguments. These are then translated by the middleware layer into an appropriate SQL query:

```
# Now insert a new person
my $success = $people->new_person
    (first_name => "Reuven",
     last_name => "Lerner",
     country => "Israel",
     email => 'reuven@lerner.co.il',
     phone => '08-973-2225');
print "Inserted successfully" if $success;
```

Because Perl's undefined ("undef") value is automatically translated into an SQL "NULL" value, the optional columns will be filled in with NULLs, as should be the case.

Appointments

Now that we have a class that handles the people in our database, we need to create an appointments class. For now, we will only concern ourselves with inserting new appointments and retrieving today's appointments.

The design of Appointments.pm (see Listing 4) is generally similar to People.pm, particularly in the way that it opens a database connection in the constructor and closes it in the automatically invoked DESTROY method. Beyond this, however, appointments keeps no state whatsoever. It merely acts as a conduit to the database, allowing us to create new appointments and find out with whom we are meeting today.

Listing 4. Appointments.p

For example, Listing 5 contains a short program that uses Appointments.pm to create a new appointment. We must create an instance of people and another of appointments. Once we have these two objects, we can set the "current person" to be my niece ("Hadar Re'em"), dying with an error if set_current_person_by_name returns undef (indicating failure).

Listing 5. Insert-appointment.pl

Once we have successfully set the current person, we can create an appointment with that person. The format of the date and time are dictated by PostgreSQL, which accepts a variety of formats.

We can similarly retrieve today's appointments using the program in Listing 6 (print-appointments.pl). This program uses the `get_today` method, which returns a list of hash references. Note that the implementation of `get_today` uses DBI's `fetchrow_hashref` method, which is known to be significantly slower than `fetchrow_arrayref`. However, it makes life much more convenient, allowing us to do print-appointments as seen in Listing 6.

Listing 6. Print-appointments.pl

Finally, we can list all of today's appointments with a particular person with the `get_today_with_person` method. Of course, this means that we must create an instance of `people` and choose a current person using one of the methods described earlier. The implementation of `get_today_with_person` expects to receive an instance of `people` as its first user-passed parameter, allowing us to use the current person in our SQL query. The program in Listing 7 demonstrates how I can find all of today's appointments that I have with my nephew Shai.

Listing 7. Print-appointments-with-shai.pl

Designing the Objects

One of the major points of using objects in a middleware layer is the fact that they provide a layer of abstraction. So long as the interface is well defined and remains stable, the implementation can change.

However, as with all programming techniques, designing good objects can be difficult. Perl provides totally open access to an object's internals, meaning that without a good API, programmers using the object might be tempted to reach inside and work directly with the implementation. This might mean that software will break when the implementation changes—the very situation that using objects was supposed to prevent!

In addition, we want our objects' implementations to be relatively separate from each other. During my design of the `people` and `appointments` objects, I was sorely tempted to allow `Appointments` to get and use the numeric ID of the current person. But of course, doing so would violate the abstraction barrier that I created with my object. The solution, which is admittedly not as elegant as I would like, was to create the `get_current_person` method. This allows

appointments to retrieve the current user, without having to know where it comes from. In the end, of course, the return value from `get_current_person` is placed in an SQL statement and is compared with `People.person_id`, breaking the abstraction somewhat.

Finally, notice how each of the objects here contains basic logic, but does not store any state. It would be relatively simple, for example, for our people object to retrieve all of the rows from the people table, and to make them available to invoking objects from within Perl. Indeed, such a solution would significantly reduce the overhead of going to a database, and would allow us to perform manipulations in Perl, rather than turning to SQL each time.

But this solution causes many more problems than it solves. For example, what happens if we create two instances of people? Now we have two objects, each of which contains the full set of rows from the people table. If one object modifies its state, that modification will never be reflected in the second object. Worse yet, what happens if both objects modify their state before storing those changes in the database? Perhaps the database is designed to resolve such locking issues, but our Perl objects are not. Furthermore, what happens when we have 100,000 people in our people table? Reading that much data into a database client is a waste of memory, and of the high-performance data selection and manipulation routines that a database server includes.

Our objects are thus pipelines to the database, giving our web application the ability to talk to a database without having to include any SQL or knowledge of the tables' layout. The objects, by providing a standard API, make it possible to change the underlying implementation without having to announce those changes to the world.

Conclusion

This month, we looked at the reasons behind three-tiered architectures and started to look into a basic implementation of an application that uses this architecture. As you can see, we can already create small text-mode applications. Next month, we will complete our implementation, giving us the skeleton for a simple appointment book using `mod_perl/Mason` and PostgreSQL. We will also discuss issues regarding the scalability of three-tier solutions, as well as some of the pitfalls.

Resources



Reuven M. Lerner owns and manages a small consulting firm specializing in Web and Internet technologies. As you read this, he should (finally!) be finishing Core Perl, to be published by Prentice-Hall later this year. You can reach him at reuven@lerner.co.il, or at the ATF home page, <http://www.lerner.co.il/atf/>.

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

[Advanced search](#)

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

[Advanced search](#)

Music to Feed Your Soul...

Marcel Gagné

Issue #81, January 2001

Marcel composes with Linux.

Would you be so kind as to turn up the volume *un petit peu*, François? *Ah, merci*. Monsieur Debussy was in fine form when he wrote that, eh, *mon ami*? Wonderful. *C'est de la tres belle musique*, François? Ah, it is like that Englishman, Shakespeare, who despite being English, certainly got it right when he said, "If music be the food of love, play on." Of course, food requires the right wine and with *Pelléas et Mélisande* playing in the background, I sense a Medoc may well be in order, say a *Cos-d'Estournel*.

Aller, François. Please bring some up from the cellar.

Ah, mes amis. Come in. Please, sit down. François has prepared your usual tables. He will be along shortly to take your orders when he returns with the wine. We were listening to a little opera when I realized that a nice bottle of wine would wonderfully enhance such delightful music.

Music, like food and cooking, is not a destination but a lifelong journey. With Linux and a few recipes from open-source kitchens around the world, everyone can now explore the world of music from the sheer enjoyment of it to learning and performing. *Mais oui*, can you not see it? Today, a simple programmer. Tomorrow, a star upon the world's musical stage. They say that you must first walk before the running, *non*? So it is with music. The lessons come first, a little practice, practice, practice and suddenly you are at Carnegie Hall, *non*?

To get you started on this road, we must start with the basics, with some note recognition and ear-training exercises. For this part of our training, we will use **GNU Solfège**.

GNU Solfège comes from the Linux kitchen of Tom Cato Amundsen. This is a wonderful little program designed to help with “ear training.” Essentially, ear training is a set of exercises designed to help young musicians learn and identify musical intervals, pitch, rhythm and so on. Another part of ear training is sight-singing (or sight-reading). A student is provided with a small tune (or chord) written out on a musical staff that they must then sing. Learning to sight-read a tune can be done alone, but certain things require an instructor or partner who can play intervals and chords for you. This is where Solfège comes into play. By definition, the meaning of the word “Solfège” is a set of vocal exercises sung to vowel (a, o, u) or to the classic syllables, do, re, mi and so on. Solfège requires that a student learn to identify (and sing) correct notes and intervals.

If you follow the SourceForge download link from the Solfège page, you can get RPM and DEB packages, making installation a breeze. But of course, the latest source tar ball is available in gzipped format. Some prerequisites are required for Solfège to install and work properly. These include **Python 1.5.2**, the **GNOME** libraries (at greater than or equal to release 1.0.50) and a little something called **PyGNOME** (at greater than or equal to release 1.0.50). Odds are pretty good that you already have Python installed on your system. The same may be true for the GNOME libraries (especially if you run GNOME as your desktop). The PyGNOME package (aka gnome-python) is often included with later Linux distros, but that is not a guarantee. On my system, I needed to install it from its source tar ball (available from the Solfège web site):

```
tar -xzf gnome-python-1.0.53.tar.gz<\n>  
cd gnome-python-1.0.53  
make  
make install
```

Once you have this in place, you are ready to install Solfège. As I mentioned, the easiest thing to do is pull down either the RPM or Debian DEB package from SourceForge. If you need (or want) to do the build yourself, that also is quite simple:

```
tar -xzf solfege-0.7-24.tar.gz<\n>  
cd solfege-0.7.24  
make  
make install
```

To run the program, simply type **solfege** at the command prompt (see Figure 1 for Solfège in action).

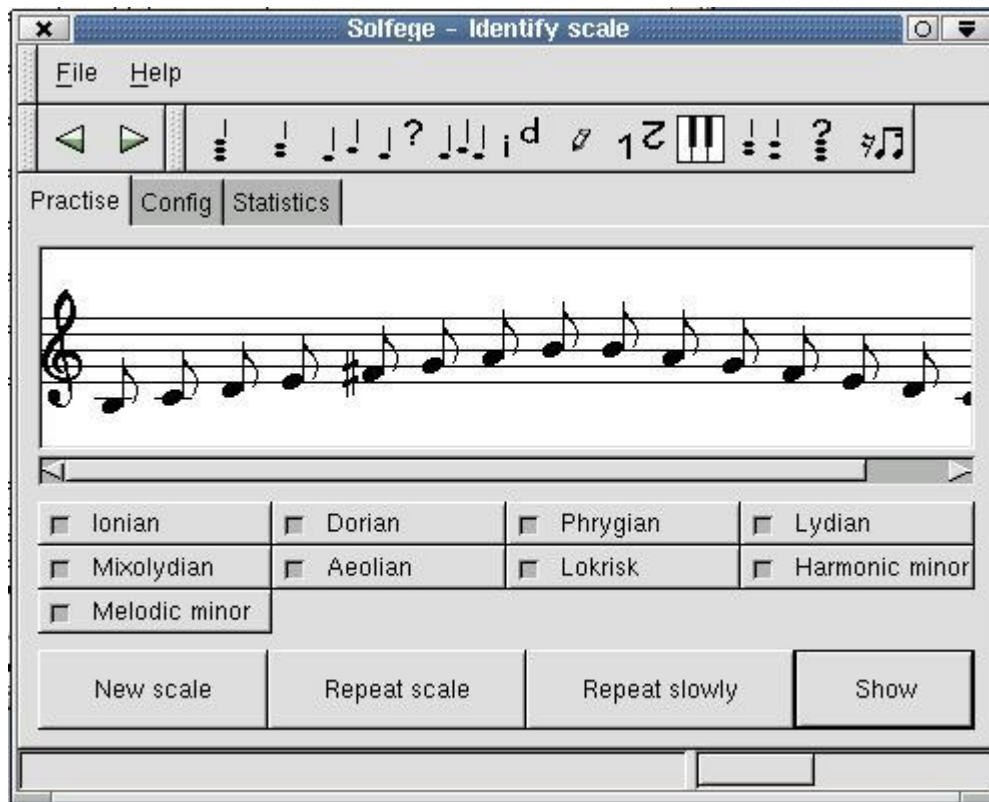


Figure 1. Learning Scales with Solfege

Now, *mes amis*, there is a small possibility that you may encounter some problems here. By default, Solfege tries to talk to your `/dev/music` device. On my system, the default MIDI device was `/dev/sequencer`, which required that François open a 1987 Musigny while I reread the INSTALL document. Not to worry though, the default device can be changed by clicking on File followed by Preferences in the Solfege interface. The next menu has a tab labeled Sound Setup where you can specify your sound device or an external player such as **Timidity**.

Ear training is a lifelong process for musicians; even seasoned professionals may find “Solfege” useful.

Those of you who are already musicians, or who aspire to be so, know the importance of rhythm and keeping time. A good metronome is one of the musician's essential tools. Alex Roberts knows this and to that end, has created **gtick**, a small GTK program that ticks off its beats in 1 (or none), 2/4, 3/4 and 4/4 time. You can set your timing from 30 to 250 beats per minute.

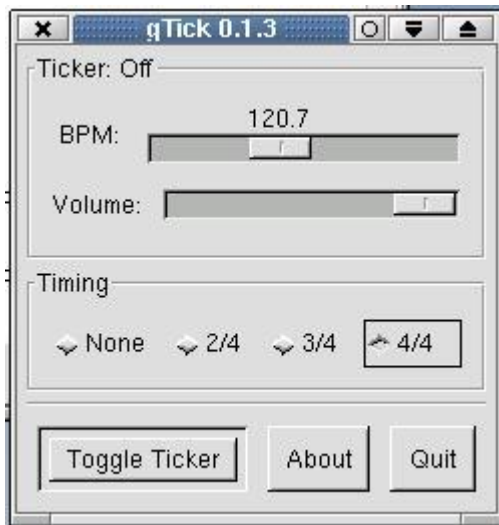


Figure 2. The gtick Metronome

You'll have no trouble building and using gtick. Just extract the source (**tar -xzvf gtick-0.1.3.tar.gz**), **cd** to the distribution directory and type **make**. The resulting binary, called gtick, can then be copied to a place where you keep executables. For instance, I copied mine into **/usr/local/bin** with:

```
cp gtick /usr/local/bin
```

Another candidate for a nice metronome is David Lee Ludwig's **gMetronome**, another GTK program. This one uses a visual, moving-beat display; eight lights bounce the beat back and forth from one side to the other. The beat itself is adjustable from 0 to 250 beats per minute. At this time, this is strictly a visual metronome rather than an audible one. Perhaps *notre ami* David would like to talk to Alex, *non*?



Figure 3. gMetronome: A Visual Beat

Like gtick, gMetronome is a breeze to compile and run. After extracting the source (**tar -xzvf gMetronome-0.1.0.tar.gz**), change directory to the newly created directory, then type **make** followed by **make install**. Type **gmetronome** and you are off.

In time, inspiration will demand that you capture the tunes you fashion in your mind so that others may share in your musical vision. With your Linux system and some great open-source tools, you can begin to set that vision in a crisp,

professional format, either as distributed MIDI files, or paper scores. For your convenience, I have already visited many of the Linux notation kitchens, and I am happy to report that we can surely find something to tempt your palate.

Forgive me, *mes amis*, but I must admit at this time that while he can sing a respectable *chanson*, your humble Chef is light years away from being a musician. Consequently, a simple point-and-click music notation system was the first thing I looked for. This brought me to **NoteEdit** written by Jörg Anders. NoteEdit is a nice-looking application with notes and rests conveniently placed on a toolbar. With NoteEdit, you can select different clefs, time and key signatures; choose between a variety of note values with a mouse click; transpose; and more. When you are convinced of your masterpiece, you can even export the final product to a MIDI file or format it for printing using MusiXTeX. For a snapshot of NoteEdit in action, have a look at Figure 4.

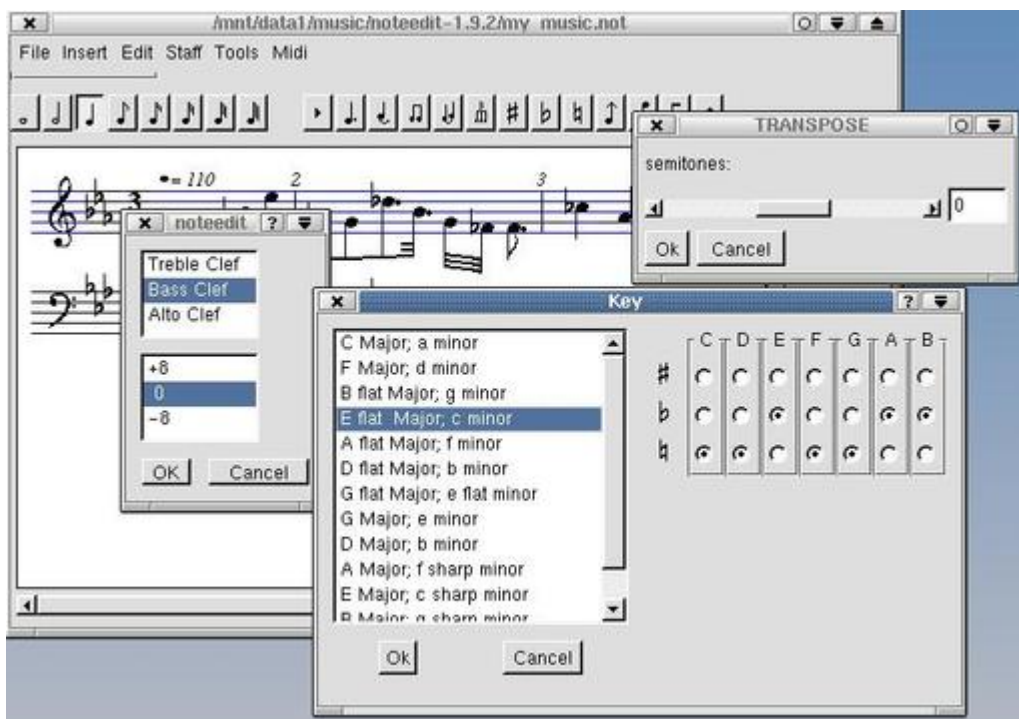


Figure 4. Getting Creative with NoteEdit

NoteEdit is a **QT** application based on the 2.x development libraries from **TrollTech**. If you are running the new KDE version 2 desktop, you will already have these on your system. To build NoteEdit, download the latest source and follow these steps:

```
tar -xzvf noteedit-1.9.2.tgz<\n>
cd noteedit-1.9.2
./configure -without-libs
make
```

Notice that I ran the **configure** script **-without-libs**. NoteEdit lets you build three different varieties of the program. The most powerful build of NoteEdit makes

use of the **TSE3** libraries. Using the libraries gives you the added capabilities of importing from a MIDI file or attached MIDI keyboard. For tonight's menu, I will concentrate on the basic recipe only. Adventurous diners should visit the NoteEdit web site for further details.

With our software built, we must first set an environment variable before we can start the program:

```
export NOTE_EDIT_HOME=/path_to/noteedit_dir<\n>  
noteedit
```

That is all. Get creative. Have fun. Pour yourself a glass of wine and let yourself go. Use the “play” button from time to time to see how you are coming along. You can also save your current masterpiece whenever you want and get back to it at a later time. Before I talk to you about printing your work, let me introduce you to one more music notation package.

Rosegarden, written by Chris Cannam, Andy Green, Richard Bown and others, is certainly worth taking a look at, though somewhat more complex than NoteEdit. As a notation editor, the package offers multiple staves, chords, beaming, triplets, slurs and a notes palette so that you can point and click your way to fame and fortune. This is also a MIDI sequencer with piano roll display. Like NoteEdit, you can also export your creation to MusixTex format for later printing.

Rosegarden can be compiled and built on Linux and a large number of other UNIX systems. You can make the installation quite easy by visiting the site and choosing one of the precompiled binaries; these are available for many platforms including Linux ELF. In order to try out Rosegarden, I downloaded the program source and built it from scratch. Once again, a straightforward build:

```
tar -xzvf rosegarden-2.1-sources.tar.gz<\n>  
cd rosegarden-2.1  
./configure  
make  
make install
```

After you have built and installed the program, you start Rosegarden by typing **rosegarden**. I used Rosegarden to import the same *magnum opus* that I was working on with NoteEdit (I know it is terrible, but I did mention that your humble chef does not usually write music). If you are feeling quite brave, you can see my work in progress in Figure 5.

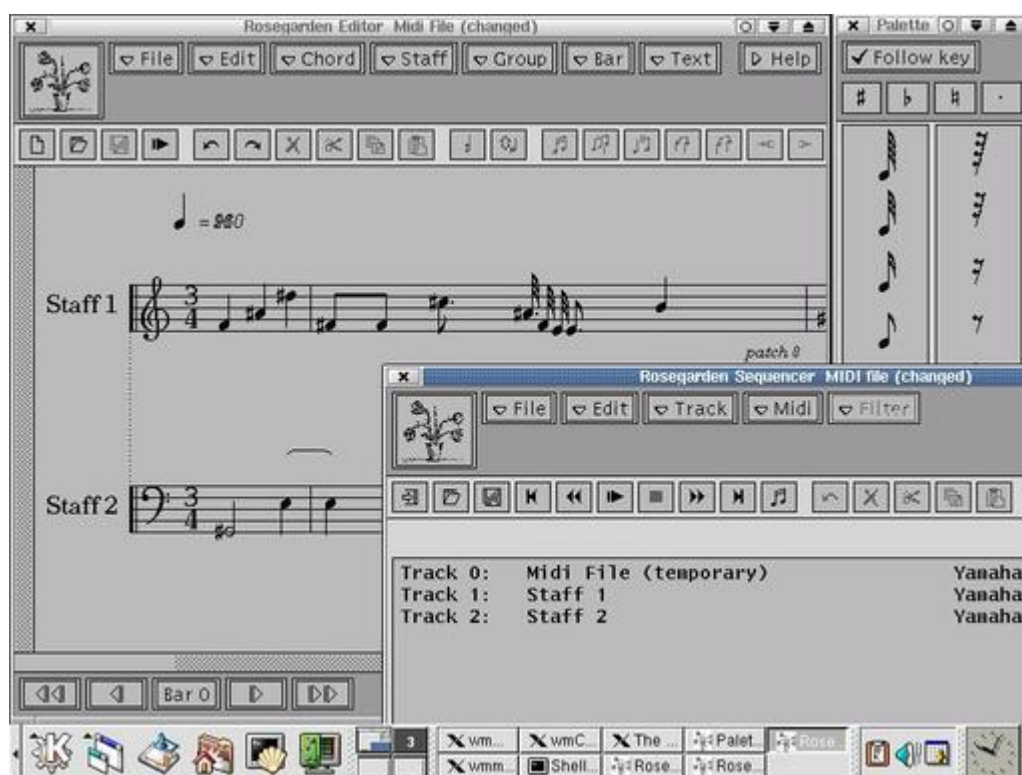


Figure 5. Making Music in the Rosegarden

Now, about printing.

Both NoteEdit and Rosegarden output their music in **MusixTex** format. MusixTex is essentially a set of extensions (fonts, programs, etc.) to the Linux **TeX** package, making it possible to print out high quality music scores. Sadly, there is no simple "Press here to print" button on any of these packages so you will have to dip into the MusixTex waters if you want to publish your music. It's not all that bad, but exploring this fully would require an article on its own. The NoteEdit web site has a nice, quick primer on working with MusixTex, and I would suggest you start there. The payoff for your hard work will be superb, high-quality printed music.

Alas, music notation for Linux is a field that is still maturing with developers heading off in many different directions. We have visual notation systems (like NoteEdit and Rosegarden) as well as text-only language software that converts your typed input to postscript pages. Some are simple, others much more complex. The beginner and the professional alike will surely find something to satisfy their needs. As François starts to clean up your plates, allow me to suggest a visit to Dave Phillips' web site where Dave is the keeper and maintainer of the "Sound and Midi Software for Linux" page. This is a great place to start looking as your musical needs grow.

Mon Dieu! It is already late, *non?* I fear *mes amis*, that we must send you all home and close Chez Marcel until next time. Since you are all within walking distance of your homes, my faithful waiter François, will top up your glasses

once again. François, pour yourself a glass as well. Let us all sit quietly as we finish up and just listen to Monsieur Debussy.

Until next time, your table will be waiting here at *Chez Marcel*.

À votre santé! Bon appétit!

Resources



Marcel Gagné lives in Mississauga, Ontario. 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 edits *TransVersions*, a science fiction, fantasy and horror magazine (now an anthology). He loves Linux and all flavors of UNIX and will even admit it in public. In fact, he is currently working on *Linux System Administration: A User's Guide*, coming soon from Addison Wesley Longman. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things from his web site at <http://www.salmar.com/marcel/>.

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

[Advanced search](#)

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

[Advanced search](#)

The 101 Uses of OpenSSH: Part I

Mick Bauer

Issue #81, January 2001

Mick scratches the surface of ssh.

It's story time here in the Paranoid Penguin. Don't worry—the story is a preface to the nuts-and-bolts sort of stuff you've come to expect in *LJ*. In fact, there are so many nuts and bolts to play with in OpenSSH that this article spills over into next month's issue!

This month we'll cover ssh's background and architecture, how to build and/or install OpenSSH, how to use ssh as an encrypted replacement for Telnet, how to set some basic ssh configuration options and how to use scp for encrypted file transfers. Next month I'll cover RSA/DSA authentication, local port-forwarding, remote-command-execution and other more advanced, and extremely powerful functions of ssh/OpenSSH.

In order to do this magnificent software justice, I'd like to begin by talking about how it got here and some of the people who brought it to us.

The Story of SSH

One of the coolest things about UNIX has been that there is not one but several different ways to administer systems from remote consoles. Sad to say, most of these methods (Telnet, rsh and X, to name a few) send everything over the network in clear text, including passwords. The combination of our reliance on the Internet with the proliferation of script kiddies and other packet-sniffing deviants has made administrative clear-text network applications obsolete.

But a few years ago Finnish über-hacker Tatu Ylonen created a mind-blowingly cool thing called the Secure Shell, or ssh. ssh is a suite of tools that roughly correspond to Sun's rsh, rcp and rlogin commands, but with one very important difference: paranoia. ssh lets you do everything rsh, rcp and rlogin do, using your choice of libertarian-grade encryption and authentication methods. But

wait—there's a catch—ssh version 1 relies heavily on RSA, an excellent, but as we say, encumbered (patented) technology that requires any application that uses it to be licensed (paid for) unless it's used in noncommercial settings (even in noncommercial use ssh's legality has always been murky, especially in the US). But wait, you say, RSA's US patents expired in September 2000—problem solved, right? Almost: Tatu's got to earn a living, so by the time RSA became less encumbered, ssh itself had become more so as his company F-Secure tightened the licensing reins. In fact, beginning with ssh version 2.0, unlicensed/free commercial use (regardless of RSA issues) was no longer permitted. All this despite Tatu's sincere desire that ssh become an Internet standard, one of the requirements of which is that at least one free implementation be available.

Enter Theo de Raadt and the OpenBSD team. OpenBSD, of course, is the ultra-secure offshoot of NetBSD, a free version of BSD UNIX. Theo and our open-source brethren in the OpenBSD project wanted to include ssh in OpenBSD 2.6 but were wary of ssh's various encumbrances. When they learned that the Swedish programmer Bjoern Groenvall had released an improved version of ssh, 1.2.12 (the last completely free-except-for-RSA version of Ylonen's ssh), the OpenBSD guys rapidly got to work on updating and adapting it for a larger audience. OpenSSH has been part of OpenBSD ever since and is now portable to most version of UNIX.

OpenSSH built on Groenvall's work (his version, called OSSH, is still available), adding support for later versions of the ssh protocol and modularizing its cryptographic mechanisms in such a way that it's possible to compile OpenSSH without any patented algorithms whatsoever (i.e., without support for ssh v.1 protocols, which depend on RSA). The other innovation the OpenBSD team brought is the forking of the OpenSSH code-base into a “clean” version, which is kept as simple and platform-independent as possible, and a “portable” version, which can be compiled for a variety of versions of UNIX besides OpenBSD.

This last innovation is of particular note to us Linux geeks: the clean version is kept that way to maximize the code's “auditability”, ensuring that it's fundamentally stable and secure. Only after this code is blessed by Theo (a righteous paranoiac) are portability enhancements added. Thus, we benefit from a software package that is both extremely secure and 100% Linux-compatible.

By the way, less than two months passed between the time the OpenBSD crew discovered OSSH and the time they released OpenSSH 1.2.2; and only six and a half months after that they released the fully-portable and ssh v.2-compatible OpenSSH 2.0. Even considering that they were building on Ylonen's and Groenvall's work, this is a remarkable achievement, especially considering the quality of the end product and the fact that nobody gets paid for it!

So that's the story of ssh and OpenSSH so far. I hope you agree that it's a pretty compelling one, as notable as OpenSSH itself, which in all likelihood will very rapidly become the preferred version of ssh for open-source versions of UNIX.

Are you all fired up about OpenSSH and ready to install it on every UNIX system you control? Good. Let's get busy!

By the way: "ssh v.1.x" and "ssh protocol v.1" refer to ssh's software-release and protocol, respectively, and are not really synonymous. But since the package and protocol major version numbers *roughly* correspond, from here on in I'll use "ssh v.1x" to refer to RSA-based versions of ssh/OpenSSH and "ssh v.2x" to refer to versions that support both RSA and DSA. And if you don't know the difference between RSA and DSA, suffice it to say that both do the same thing but DSA has no patent- or license-restrictions.

How SSH Works

Secure Shell works very similarly to Secure Sockets Layer web transactions (it's no coincidence that the cryptographical functions used by OpenSSH are provided by OpenSSL, a free version of Netscape's Secure Sockets Layer source-code libraries). Both can set up encrypted channels using generic "host keys" or with published credentials (digital certificates) that can be verified by a trusted certificate authority (such as VeriSign). Here's how connections are built.

First, the client and the server exchange (public) host keys. If the client machine has never encountered a given public key before, both ssh and most web browsers ask the user whether to accept the untrusted key. Next, they use these to negotiate a session key that is used to encrypt all subsequent session data via a block cipher such as Triple-DES (3DES), blowfish, or idea.

Then, the server attempts to authenticate the client using RSA or DSA certificates. If this isn't possible, the client is prompted for a standard username/password combination (optionally, "rhosts" host-IP-based authentication with or without RSA keys may be used; OpenSSH also supports KerberosIV and skey). Finally, after successful authentication the session proper begins: either a remote shell, a secure file transfer, a remote command, etc., is begun over the encrypted tunnel.

As I mentioned earlier, ssh is actually a suite of tools:

- sshd—dæmon that acts as a server to all other commands
- ssh—primary end-user tool: remote shell, remote command, and port-forwarding sessions

- scp—tool for automated file transfers
- sftp—tool for interactive file transfers—COMMERCIAL SSH ONLY
- ssh-keygen—generates private-public key pairs for use in RSA and DSA authentication (including host keys)
- ssh-agent—daemon used to automate client's RSA/DSA authentications
- ssh-add—loads private keys into ssh-agent process
- ssh-askpass—X interface for ssh-add

Note that sftp, which is essentially an ftp client with encryption and strong authentication grafted on, is available only in F-Secure's commercial versions of ssh version 2—I only mention it here because you may come across a reference to it elsewhere and wonder why you've only got scp.

Of these tools, most users concern themselves only with ssh, since “encrypted Telnet” is the simplest use of ssh. Scp, ssh-agent and ssh-add, however, along with the strong authentication and TCP port-forwarding capabilities of ssh itself, make ssh considerably more flexible than that. Since we're paranoid and want to encrypt as much of the stuff we fling over the networks as possible, we really groove on this flexibility.

Getting and Installing OpenSSH

The OpenSSH web site (see Resources) is the place to go for the latest version of OpenSSH, both in source-code and RPM forms, and also for OpenSSL, which is required by OpenSSH. Also required is zlib, available at the freesoftware.com site (see Resources).

Note that you may not be able to get by with RPM packages. When I tried to install the RPMs from OpenSSH.com on my laptop, running SuSE Linux, everything worked except sshd, which wouldn't install due to SuSE's lack of a “chkconfig” package. Your preferred flavor of Linux may or may not have the same problem (unless it has chkconfig), or it may have its own RPM of OpenSSH (for all I know, by the time you read this somebody may have published RPMs for SuSE).

If RPMs won't work, you'll need to build OpenSSH (and possibly OpenSSL and zlib) from source. To Linux old-timers, “rolling your own” software installations is old hat, but if you're not in that category don't despair. All three distributions use “.configure” scripts that eliminate the need for most users to edit any Makefiles. Assuming your system has gcc and the normal assortment of system libraries and that these are reasonably up-to-date, the build process is both fast and simple.

In my own case, after installing OpenSSL 0.9.5a and zlib-1.1.3 (all version numbers, by the way, may be outdated by the time you read this!) I followed these steps to build and install OpenSSH 2.1.1p4:

```
tar -xzvf openssh-2.1.1p4.tar.gz
cd openssh-2.1.1p4
./configure --sysconfdir=/etc/ssh
make
make install
```

Per instructions provided by the file "INSTALL" I fed the configure script one customized option: rather than installing all configuration-files in **/etc**, I instructed it to create and use a subdirectory, **/etc/sshd**. Since this version of OpenSSH supports both RSA and DSA keys, it makes sense to minimize the amount of clutter ssh adds to **/etc**.

For a client-only installation, this is all you need to do. Note that one or more of the version numbers cited above may already be dated by the time you read this article; be sure to check the OpenSSH and zlib web sites for the latest versions.

If you wish to run the Secure Shell Dæmon sshd (i.e., you wish to accept ssh connections from remote hosts), you'll also need to create startup scripts and, in the case of SuSE, edit **/etc/rc.config**. This has also been thought of for you: the source distribution's "contrib" directory contains some useful goodies.

The Red Hat directory contains "sshd.init", which can be copied to **/etc/rc.d** and linked to in the appropriate runlevel directory (**/etc/rc.d/rc2.d**, etc.). It also contains "sshd.pam", which can be installed in **/etc/pam** if you use Pluggable Authentication Modules (PAM) and "openssh.spec", which can be used to create your very own OpenSSH RPM package. These files are, obviously, intended for use on Red Hat systems but will probably also work on Red Hat-derived systems (Mandrake, Yellow Dog, etc.).

The suse directory also contains an "openssh.spec" file for creating OpenSSH prpm packages for SuSE and an "rc.sshd" file to install in **/etc/rc.d** (actually **/sbin/init.d** in SuSE). In addition, it contains "rc.config.ssd", the contents of which must be added to **/etc/rc.config** in order for the rc.sshd script to work properly. This is achieved by simply entering the command:

```
cat ./rc.config.ssd >> /etc/rc.config
```

Create a symbolic link in rc2.d and/or rc3.d, and your SuSE system is ready to serve up secured shells! Either reboot or type **/etc/rc.d/rc.sshd start** to start the dæmon.

SSH for the Masses: Doing the “Encrypted Telnet” Thing

What if all you need are interactive shell sessions on remote systems *à la* Telnet? Chances are that even without so much as looking at a configuration file, you need to simply enter:

```
ssh remote.host.net
```

You will be prompted for a password (ssh will assume you wish to use the same user name on the remote system as the one you're currently logged in with locally), and if that succeeds, you're in! That's arguably simpler and indisputably much more secure than Telnet.

If you need to use a different user name on the remote system than the one you're logged in with locally, you need to add the flag `-l` followed by your remote username. For example, if I'm logged on to my laptop as “mick” and wish to ssh to kong-fu.mutantmonkeys.org as user “mbauer”, I'll use the command:

```
ssh -l mbauer kong-fu.mutantmonkeys.org
```

What is this doing for me? Nothing seems much different from Telnet. I may be asked whether to accept the remote server's public key, it may take somewhat longer for the session to get started and depending on network conditions, server load, etc., the session may seem slightly slower than Telnet, but for the most part I won't notice much difference.

But I will have logged in without sending my username and password over the network in clear text, and all session data will be encrypted as well. I can do whatever I need to do, including `su -`, without worrying about eavesdroppers. And all it costs me is a tiny bit of latency!

Digging into Configuration Files

Configuring OpenSSH isn't complicated either. To control the behavior of the ssh client and server there are only two files to edit: `ssh_config` and `sshd_config`. Depending on the package you installed or the build you created, these files are either in `/etc` or some other place you specified using `.configure's` `-sysconfdir` directory.

`ssh_config` is a global configuration file for ssh client sessions initiated from the local host. Its settings are overridden by command-line options and by users' individual configuration files (kept, if they exist, in `$HOME/.ssh/config`). For example, if `/etc/ssh/ssh_config` contains the line:

```
Compression no
```

but the file `/home/bobo/.ssh/config` contains the line

```
Compression yes
```

then whenever the user “bobo” initiates a ssh session, compression will be enabled by default, even though for users without this setting in their own `$HOME/.ssh/config` files compression will be turned off. If, on the other hand, bobo invokes ssh with the command:

```
ssh -o Compression=no remote.host.net
```

then compression will not be enabled for that session.

In other words, the order of precedence for ssh options is, in decreasing order, the ssh command-line invocation, `$HOME/.ssh/config`, and `/etc/ssh/ssh_config`.

`ssh_config` consists of a list of parameters, one line per parameter, in the format:

```
parameter value(s)
```

Some parameters are Boolean and can have a value of either “yes” or “no”. Others can have a list of values separated by commas. Most parameters are self-explanatory, and all are explained in the `ssh(1)` man page. Table 1 shows a few of the most useful and/or important ones (italicized text indicates possible values).

Table 1. Some Basic Client Options for `ssh_config`

There are many other options in addition to these; some of them will be covered in Part II of this article. Refer to the `ssh` man page for a complete list.

Configuring and Running `sshd`, the Secure Shell Dæmon

All of that's fine if the hosts you connect to are administered by other people. But we haven't yet talked about configuring your own host to accept ssh connections. As it happens, this is very simple.

As with the `ssh` client, `sshd`'s default behavior is configured in a single file, “`sshd_config`” that resides either in `/etc` or wherever else you specified the `ssh`'s configuration directory. And as with the `ssh` client, settings in its configuration file are overridden by command-line arguments. Unlike the `ssh` client, however, there are no configuration files for the dæmon in individual users' home directories; ordinary users can't dictate how the dæmon behaves.

Table 2 describes a few of the things that can be set in `sshd_config`.

Table 2. Setting Parameters in sshd_config

There are many other parameters that can be set in `sshd_config`. We'll cover some of those next month, but understanding the above is enough to get started (assuming your immediate need is to replace Telnet and ftp).

Using scp for Encrypted File Transfers

We'll cover one more ssh topic before we adjourn for this month. The `scp` command, in most ways equivalent to the old `rcp` utility, is used to copy a file or directory from one host to another. (In fact, `scp` is based on `rcp`'s source code.) In case you're unfamiliar with either, they're noninteractive: each is invoked with a single command line, in which you must specify the names and paths of both of what you're copying and where you want it to go.

This noninteractive quality makes `scp` a bit less user-friendly than `ftp`: like it or not, to use `scp` you need to read its man page (or articles like this) and memorize a few flags. But like most other command-line utilities, `scp` is far more useful in scripts than interactive tools tend to be. Using `scp` “on the fly”, though, is easy to learn. The basic syntax of the `scp` command is

```
scp [ options ] sourcefilestring destfilestring
```

where the source and destination file strings can either be a normal UNIX file/path string (e.g., `./docs/hello.txt`, `/home/me/mydoc.txt`, etc.) or a host-specific string in the format

```
username@remote.host.name:path/filename
```

For example, suppose you're logged into the host “`crueller`” and want to transfer the file “`recipe`” to your home directory on the remote host “`kolach`”. Suppose further that you've got the same user name on both systems. The session would look something like this (user input in bold):

```
crueller: > scp ./recipe kolach:~  
mick@kolach's password: *****  
recipe          100% |*****>| 13226    00:00  
crueller: >
```

After typing the `scp` command line, we were prompted for our password (our username, since we didn't specify one, was automatically submitted for us using the username we're logged on to `crueller` as). `scp` then copied the file over, showing us a handy progress bar as it went along. And that's it!

Suppose you're logged on to `crueller` as “`mick`” but have the username “`mbauer`” on `kolach` and wish to write the file to `kolach`'s directory **`data/recipes/pastries`**. Then our command line would look like this:

```
crueller: > scp ./recipe mbauer@kolach:/data/recipes/pastries/
```

Now let's switch things around. Suppose we want to retrieve the file **/etc/oven.conf** from kolach (we're still logged in to crueller). Then our command line looks like this:

```
crueller: > scp mbauer@kolach:/etc/oven.conf .
```

Get the picture? The important thing to remember is that the source must come before the destination.

Although most users use ssh and scp for simple logins and file transfers, respectively, this only scratches the surface of what ssh can do. Next month we'll examine how RSA and DSA keys can be used to make ssh transactions even more secure, how "null-passphrase" keys can allow ssh commands to be included in scripts, how to cache ssh credentials in RAM to avoid unnecessary authentication prompts and how to tunnel other TCP services through an encrypted ssh connection.

Resources



Mick Bauer (mick@visi.com) is security practice lead at the Minneapolis bureau of ENRGI, a network engineering and consulting firm. He's been a Linux devotee since 1995 and an OpenBSD zealot since 1997, taking particular pleasure in getting these cutting-edge operating systems to run on obsolete junk. Mick welcomes questions, comments and greetings.

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

[Advanced search](#)

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

[Advanced search](#)

Home Network Push Accelerates

Linley Gwennap

Issue #81, January 2001

Linley explores the potential for Home Networks.

Although home networks are merely geek chic today, new products are emerging to help drive them into the mainstream. These new devices, many based on Linux, will drive an explosion of interest in home networks in affluent, but not necessarily tech-oriented, households. As in any revolution, however, various players are each pushing their own visions of how the technology will work, creating conflicting information for potential customers.

Many important vendors recently created the Internet Home Alliance to help promote home networks. These vendors include diverse players such as networking giants Cisco Systems, Sun Microsystems and 3Com; consumer-focused companies Best Buy, CompUSA, Panasonic and Sears; and chip makers Motorola and Texas Instruments. Although these vendors are working together, each of them is trying to solve a particular problem: sharing a broadband connection, for example, or creating a music network (see "Linley on Linux", page 186 of October 2000). Indeed, many early efforts will be single-purpose networks driven by a particular vendor's solution. Once home networking becomes established, however, there will be many new uses. For example, the TiVo DVR (digital TiVo video recorder) does an excellent job of storing TV programs on its hard drive for later viewing. But there is no practical way to record programs between two televisions in different rooms. A home network would allow a single DVR to drive video streams to multiple televisions.

Ultimately, I expect the home network to look much like the network in a small office, with a single server connected to a number of clients. The server will have two main components: the residential gateway and the storage server.

Many vendors are touting their expertise in residential gateways, but few are available today. These devices are similar to WAN gateways, connecting the internal home LAN to the Internet, typically through a broadband connection

such as DSL or cable modem. Unlike a standard broadband modem, however, the residential gateway has a LAN interface to distribute data to multiple clients.

The storage server will consist of a processor, network connection and one or more hard drives, providing data storage for all devices in the home. This server could back up the hard drives of any PC in the home; store digital photographs, music and video; and stream audio or video content to any network client. It should be programmable enough that it can host applications, such as TiVo, that autonomously obtain broadband content or perform other housekeeping functions.

Client devices include PCs, Macs and Linux systems that use the residential gateway to access the Internet and use the storage server to back up important files or simply as shared data storage (ideally, the storage server should use a RAID design so the inevitable hard-drive failure does not eliminate vital data). Photographs or music files can be created on a PC or downloaded from the Internet, then transferred to the storage server.

Once on the storage server, a photograph can be displayed on any TV screen or monitor in the house via a simple video client. A video client has only a processor, a network interface and a video output. The client downloads an image from the storage server, decompresses it and displays it. The video client also downloads and displays a video stream from the server. Similarly, an audio client connected to an amplifier and speakers can play any music file on the storage server. The audio client has a similar bill of materials as the video client. Either client should sell for less than \$100 in high volume, making it cost-effective to have several throughout the house.

Other potential network clients are Internet appliances, non-PC devices that have a web browser and e-mail. Video games connect to the network to enable multiplayer gaming. Even appliances such as refrigerators and thermostats might connect to the home network to share usage information and operate more efficiently.

This architecture centralizes the storage and main compute resources in the storage server, where disk space can be shared efficiently and upgraded easily using internal or external drives—"Honey, can you pick up a few gigabytes on the way home?" Initially, single-function products such as TiVo and AudioReQuest will employ their own hard drives, but these units will ultimately give way to low-cost thin client devices.

Linux is likely to play a key role in this growing network of consumer products. The storage server in particular needs a robust but inexpensive operating

system with built-in networking capabilities. Since the storage server is likely to run more complicated applications that organize and manage audio, video and web content, it needs a platform with good software-development tools and standards. Linux is an ideal fit.

The residential gateway may be a simple networking device, but it is likely to run a firewall and perhaps other software as well. Linux may play a role here. Some of the client devices may run embedded Linux as well; the key issue here is maintaining a small memory footprint for reduced cost. Certainly, Linux is a good choice for Internet appliances with web browsers.

One concern with home networks is the choice of physical connection. The most cost-effective choice today is to use a home's existing phone wiring (HPNA). I don't see this as a good long-term solution, because most homes, particularly outside of the US, don't have enough phone jacks. The long-term winner is likely to be a wireless solution such as 802.11. The 802.11b standard provides enough bandwidth for several audio streams but only one video stream. The forthcoming 802.11a version is needed for multiple simultaneous video streams.

In the short-term, these two media, as well as power-line networking (HPPA), will slug it out, creating confusion in the market. The solution is to support two or all three standards in the residential gateway, creating a mixed-mode LAN compatible with clients of different types. This method will increase the cost of the gateway but ease consumer concerns.

As these problems are resolved and costs begin to drop, home networks will flourish. Some studies project that more than one-third of US homes will have a broadband Internet connection by 2004. These homes will be prime candidates for home networks. As music, photos and video all become compressed digital files, the advantages of a home network will become clear. The growth of home networks will create opportunities for new Linux-based devices, such as residential gateways and storage servers.



Linley Gwennap (linleyg@linleygroup.com) is the founder and principal analyst of The Linley Group (<http://www.linleygroup.com/>) analysis firm in Mt. View, California.

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

[Advanced search](#)

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

[Advanced search](#)

Focus on Software

David A. Bandel

Issue #81, January 2001

Simple Document Management System, djpim, Internal People Tracking System and more.

I sure see a lot of requests to do things the Microsoft Way, but I see little sense in it. A number of recent converts want to get away from Microsoft for whatever reason, but then expect everything to be done in the Microsoft way. I would never even *mention* Microsoft except as a bad example. Now I hear a cry for a Linux registry. Good grief, Charlie Brown! This is one of the elements in Windows that causes more problems than it solves. Why move simple, easy-to-read and edit (okay, not always so simple to read) files out of /etc and cram them into one large file that requires special tools to edit? I vividly remember a promise that the Windows registry would never have to be edited, that it was only for use by the system. To date, I have never seen a Windows system that did not need the registry edited. But I guarantee the learning curve on the files in /etc is a lot less steep than the Windows registry. So before screaming for changes, why not make sure that whatever changes are proposed are changes for the better? Linux needs to improve, but a Linux registry will not qualify as an improvement.

Simple Document Management System: <http://sdms.cafuego.net/>

This simple document management system is exactly what it says it is: simple. It is simple to install and simple to use. I've seen other document management systems, and while this one does not seem to have all the flair of some others, it works exceedingly well. The nicest part is that it uses a web browser. That document you need that's halfway around the world is now accessible. Of course, if it's a sensitive document, you'll want to use a secure web server. It uses ACLs, so you can restrict who can do what with the documents. Requires: MySQL, web server with PHP4 and MySQL support, a web browser.

djpim: <http://www.tcomeng.com/djpim/>

This utility, called Daryl Jones' Personal Information Manager (djpim), is a well-implemented web-based information manager. It can be used for a number of things, including tracking projects within a department. As a "Personal Information Manager" it lacks an integrated calendar. You can pop up a small calendar in two places, but it's not quite the same. Other than that, if you need to be reminded of a list of tasks, this utility is aesthetically pleasing and well organized. Requires: Web server w/PHP and MySQL support, MySQL, a web browser.

Internal People Tracking System: <http://dev.wslogic.com/~anderson/ipts/>

If you have folks scattered all over town, or worse, all over the state, you can keep track of them with this little tool. You might want to add or change some of the IN/OUT information provided in the web page, but that's easily done. The most difficult part of using IPTS is going to be making the employees use it. Requires: MySQL, web server w/PHP and MySQL support, web browser, and Perl modules: HTML::Template, CGI_Lite, DBI, and DBD::mysql.

pad: <http://www.lammah.com/pad/>

If you have very sensitive data, you can use this utility to break up the data into multiple, encrypted files. By putting each resulting file on a different server and telling someone where to find them, only someone with access to all the files can reassemble them. No password is required to encrypt or decrypt, but it is necessary to have all the required files intact. The only disadvantage is that your storage requirements will, at a minimum, double. Requires: libcrypt, libm, libc, libdl.

SafetyNet: <http://unixpimps.org/safetynet/>

This shell script makes sure the services you want running at all times continue to run. If they are not running, it restarts them and sends you an e-mail. No more wondering if all the services you need are running or not. If, for some reason, the service can't be restarted, you'll see that in the e-mail message as well. Requires: a Bourne shell, recommends cron.

slmon: <http://www.m00se.hsn.pl/slmon/>

Just what we needed, another system performance monitor. But wait, this one is a little different, and you might want to take a second look at it. It boasts a couple different modes and supports multiple CPUs for those fortunate enough to have them. You can see a graph showing each CPU and its memory or view a histogram of each CPU's load one at a time. Requires: libslang, glibc, libdl, libm.

GtkPortFolio: <http://www.centercube.com/GtkPortFolio/>

For all you market wizards out there, GtkPortFolio is a very well-done quote downloader. It's easy to use. This is what is really better termed userware. Want to add a symbol? No need to edit files, just put the symbol in the Add Symbol box, and it will be remembered. Don't know a symbol? Put the company's name in the search box, and it will open Netscape and show you the symbol(s) it found. It's about as simple as it comes. Requires: Perl, perl modules: Gtk, Gtk::Gdk::ImlibImage, Finance::Quote, Finance::YahooChart, Time::localtime, LWP::Simple.

nscache: <http://nscache.sourceforge.net/>

If you've ever used Netscape, you know how large the cache can become. Or maybe you don't, but let me tell you it can get *very* large. This utility allows you to use a GTK GUI to browse through your Netscape cache. You can select from two views: tree view or sorted view. The sorted view allows you to see the various entries sorted by URL (default), size, access time or mime type. Both views allow you to delete any file in the list. Now's your chance to dump some of those really large cached files. Hit the sorted view, sort by size, scroll down to the bottom, right click on your largest entries, select delete and recoup some space. Requires: libdb, libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.

Until next month.



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

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

[Advanced search](#)

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

[Advanced search](#)

The Tip of the Embedded Linux Iceberg

Rick Lehrbaum

Issue #81, January 2001

"Show me the Linux!" you say? Embedded Linux-based technology is closer than you think.

People keep asking, "All this talk about Embedded Linux taking off like a rocket sounds great, but are any real companies using Embedded Linux in real products? And, if so, when do these Embedded Linux-based products start hitting the market?"

The answer is "You bet they're designing Embedded Linux into real products! Lots of 'em!" As for when these products start shipping to customers, some are already being delivered in large quantities, and many more are in varying stages of development. Bear in mind, the gestation cycle of most new product-development projects is roughly nine to twelve months, whereas Embedded Linux itself has only just begun to see widespread interest in the past twelve months. If you do the math based on those two assumptions, you'll quickly realize that the rollout of Embedded Linux-based products ought to be starting right about now and is expected to pick up momentum toward the end of this year.

Although *Linux Journal* and others have covered a number of consumer-oriented, Embedded Linux-based devices through news announcements, preview stories and product reviews, many design projects are, for obvious reasons, conducted beneath a cloak of secrecy. As you can imagine, suppliers of Embedded Linux are fond of saying, "We really wish we could tell you about some of the exciting products we've been designed into, but we're under nondisclosures that prevent us from talking about them."

Here, then, is a brief rundown on many of the Embedded Linux-based, consumer-oriented devices that have been disclosed publicly. Bear in mind that the list below represents merely the tip of the Embedded Linux iceberg, as it were.

PDAs

Samsung Yopy PDA--The Yopy offers a browser, MP3 player and CompactFlash in a compact PDA-sized package. The device has a 3.9" backlit color LCD screen, uses an ARM processor and provides both RS232 serial and USB expansion interfaces. All "standard" PDA applications are included. See details at www.linuxdevices.com/news/NS6600527506.html.



Figure 1. Samsung Yopy PDA

Agenda VR3 PDA--The VR3 is a full-function PDA with a 160x240 pixel (2.25"x3.25" viewable area) backlit LCD. It is based on a 66MHz 32-bit NEC VR4181 processor, and it has 8MB of system RAM and up to 8MB of built-in flash storage. The built-in flash memory prevents data loss due to a discharged battery condition. The device has a standard RS232 serial port plus a special high-speed serial port, along with an IrDA interface. Its operating system is Linux-VR. See details at www.linuxdevices.com/articles/AT4992223978.html.



Figure 2. Agenda VR3 PDA

Compaq iPAQ PDA--Though not natively equipped with Embedded Linux, multiple projects are underway to develop Linux implementations for the iPAQ, including one from the Compaq-sponsored handhelds.org site. The iPAQ has a 240x320 pixel backlit color LCD screen and is powered by a 206MHz Intel StrongArm processor with 32MB of RAM and 16MB of flash memory. External

interfacing and expansion are via IrDA, serial (sync/async), USB and PCMCIA. See details at www.handhelds.org/Compaq/iPAQH3600/iPAQ_H3600.html.



Figure 3. Compaq iPAQ PDA

Phones of Various Shapes and Sizes

Aplio/PRO Internet Phone--Aplio's Internet phone is a compact speakerphone-like appliance that contains a tiny embedded Linux computer running on an Aplio/TRIO system-on-chip processor. Internal memory consists of 4MB of RAM plus a 2MB flash disk, and the Internet connection is made via either a built-in modem or Ethernet (hub) function depending on model. The operating system is derived from uClinux. See details at www.linuxdevices.com/articles/AT9173372049.html and in "VoIP and Embedded Linux" (*LJ*, September 2000)



Figure 4. Aplio/PRO

Ericsson Cordless Screen Phone--The device is basically a wireless webpad with a built-in telephone and Bluetooth wireless technology for in-home use. It can surf the Web, check e-mail, send voice clips and make phone calls. The embedded computer is based on a National Geode CPU running a Red Hat supplied Linux operating system. The GUI framework is derived from Trolltech's Qt/Embedded GUI toolkit, and the browser is from Opera. See details at www.linuxdevices.com/articles/AT4268573160.html.



Figure 5. Ericsson Cordless Screen Phone

SK Telecom IMT2000 WebPhone--This combination cell phone and PDA has a 4" LCD screen and a built-in video camera. It looks like a PDA but includes a CDMA cell-phone module inside its case. Based on a StrongARM SA1110 206MHz CPU with 32MB of RAM, plus up to 32MB internal flash memory, the operating system software is PalmPalm's Tynux embedded Linux with Qt/Embedded for GUI support plus Opera's browser. See details at www.linuxdevices.com/articles/AT3334419107.html.

TV Set-Top Devices

Nokia Media Terminal Set-Top Box--This device adds a broad range of Internet-based services to your normal TV set. Among the services available are digital audio/video, digital TV, video-on-demand, cached TV programs, web access, e-mail and chatting, games/gaming and many web-based software applications. The Media Terminal's internal software is based on Linux, the Mozilla open-source browser, X Window System, plus a unique user-interface technology called "Nokia Navi bars". The embedded computer is a 366MHz Intel Celeron CPU with the Intel 810 chipset, plus 32MB SDRAM and a minimum 20GB hard disk. See details at www.linuxdevices.com/articles/AT4370516520.html.



Figure 6. Nokia Media Terminal

Indrema Set-Top Gaming System--The Indrema Entertainment System (IES) is a set-top device that converts a TV into a high-end gaming system. The device is packaged in a sleek enclosure with the look and feel of a top-of-the-line VCR. It offers a choice of dialup or broadband Internet access (it has a built-in 10/100MB Ethernet controller) and comes with a game controller. The embedded computer is based on a high-end "X86" processor, plus 64MB RAM and up to a 50GB hard disk. High-speed graphics are generated by means of an

upgradeable plug-in module, which initially uses an an NVIDIA GPU. The embedded operating system is DV Linux, an open-source Linux port targeted to gaming systems. See details at www.linuxdevices.com/articles/AT2772260294.html.



Figure 7. Indrem Set-Top Gaming System

TIVO--The TiVO “personal video recorder” represents what may well be the most well-known Embedded Linux system in existence. It also represents one of the first consumer appliances to make use of Embedded Linux. The device is based on a 54MHz PowerPC 403GCX processor with 16MB of RAM, plus a large capacity hard disk for up to 30 hours of TV program storage. Surprisingly, video is not generated in a computer-like manner but is instead based on a graphics rendering chip. Hence, the device contains a home-grown port of Embedded Linux but neither uses nor requires either a windowing system or GUI framework. See details at www.linuxdevices.com/news/NS8858229837.html.



Figure 8. TiVo

Webpads

FrontPath ProGear Wireless Webpad--A wireless Linux-based portable information appliance targeted to vertical market segments. The device supports various rich media formats, includes a 10.4" TFT display and obtains user input from either a virtual keyboard or handwriting recognition on its touchscreen. See details at www.linuxdevices.com/articles/AT5771747599.html.



Figure 9. FrontPath ProGear Wireless Webpad

Screen Media FreePad--An easy-to-use, full-featured webpad that provides a complete set of communications and computing functions including web browsing, e-mail, telephone and answering machine functions, PDA functions and Smartcard terminal functions. The device has a large (10.4") LCD screen, touch input, built-in DECT wireless technology and provides a USB interface for external (wired) expansion. The processor is a 166 MHz MediaGX with 32MB internal RAM plus a 16MB internal flash disk. The embedded operating system is Linux, the windowing system is Nano-X from the Microwindows project and the browser is from Opera. See details at www.linuxdevices.com/articles/AT2655123453.html.



Figure 10. Screen Media FreePad

Audio Entertainment Devices

Kerbango Internet Radio--Connected to the Internet via a phone line, Ethernet or a USB-connected networking interface, Kerbango Internet Radio's internal embedded computer is based on an 80MMHz Motorola PowerPC system-on-chip processor along with 8MB of RAM and 8MB of flash storage memory. The Embedded Linux operating system is based on MontaVista Hard Hat Linux with the addition of the unique Kerbango Audio Operating System (KAOS), which provides easy-to-use features for user interface, network setup, etc. In addition to "receiving" radio stations via the Internet connection, the device includes a conventional FM radio receiver and built-in FM antenna. See details at music.gamespot.com/features/kerbango.



Figure 11. Kerbango Internet Radio

Diamond Rio Audio Receiver--This home audio receiver, which won ZDNet's first-ever "Tech Trendsetter" award, enables consumers to stream MP3 music from a PC to any room in the house, giving consumers access to MP3 music via their living room stereo where they were previously limited to portable digital audio players. Its embedded computer is based on a Cirrus Maverick system-on-chip processor running Embedded Linux. See details at www.linuxdevices.com/news/NS7845657816.html.



Figure 12. Diamond Rio Audio Receiver

PhatNoise PhatBox Car MP3 System--The PhatBox car audio system, which won "best overall product" at the Third Annual MP3 Summit, gives consumers the capability to take MP3 music files from their PC and listen to them in the sound system of their car. Its embedded computer is based on a 74MHz Cirrus Logic EP7212 system-on-chip processor, running Embedded Linux. See details at www.linuxdevices.com/news/NS7845657816.html.



Figure 13. PhatNoise PhatBox Car MP3 System

empeg Car MP3 System--The empeg car (now in its second revision) is an in-dash digital music player with a capacity of up to 600 hours of high-quality music. Connected to a PC to receive downloaded MP3 files via USB, Ethernet, or a serial port, and can store over 600 hours of music on a pair of internal laptop-style hard drives. Inside, there's a 220MHz Intel StrongARM system-on-chip processor, 12MB RAM, and 1MB flash for bootstrap. See details at www.linuxdevices.com/articles/AT5630105143.html.



Figure 14. empeg Car MP3 System

Internet Access Terminals and Appliances

Adomo Home Information Appliance--The AdomoWing is part of an Adomo home information system product line, comprised of distributed devices plus a server/gateway device. The AdomoWing is basically a thin-client computer system built on a 90MHz Motorola Coldfire system-on-chip processor, a VGA video controller, several programmable status LEDs, connections for infrared keyboard/mouse, PS/2 mouse, microphone, speaker and a wireless LAN interface. It runs a variant of uClinux along with Tiny-X (XF86 based). See details at www.linuxdevices.com/articles/AT5077748575.html.



Figure 15. And More to Come

With new Embedded Linux-based devices being unveiled continually, LinuxDevices.com will maintain a continuously updated on-line "Linux-based Devices Quick Reference Guide". Check out the latest up-to-the-minute roundup at: www.linuxdevices.com/articles/AT4936596231.html.



Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com “embedded Linux portal”, which recently became part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in launching the Embedded Linux Consortium.

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

[Advanced search](#)

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

[Advanced search](#)

Anachronisms Forever

Stan Kelly-Bootle

Issue #81, January 2001

This column is currently busy serving other more-deserving readers.

This column is currently busy serving other more-deserving readers. Please wait while we play Wagner's *Ring Cycle*—or you can piss off quick—who needs *you* Philistines? You can press #1 for Solti (the default), #2 for Karajan or #3 for Sir Andrew Lloyd Webber. Be warned: #3 is the gotchyer exit that precludes all future re-entry.

We'll “hangup” on you, reminding us that the anachronism has survived wondrously from Homer's misplaced metallurgy (according to Robert Graves, Homer nodded over his iron and bronze age technologies), through Shakespeare's chiming clocks (*Julius Caesar*), and into our post-vertical ebonite telephony.

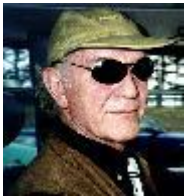
We no longer “hangup” the phone, simply bang the bugger down flat. Yet there's still a BEL lurking in the ASCII char-set although gongs per se have been replaced by tones—and even Ma Bell, my ever-pined heartthrob (I had her once in the back of a London taxi), has deserted wire-ridden communications for cable TV and credit cards.

Simple verbs such as “ship” have also suffered the anachronistic “curse”. When X says they'll “ship” Y by Z, or else, I can't help but think of the *Titanic*, *Lusatania*, *Bismarck* and *Scharnhorst*.

Digging deeper, dear patient reader, Virgil and that crowd, predicted the near-future subjunctive gerundive (whatever) with *in navem impositurus esse* (to be about to [load] ship), but with the ironical secondary meaning of *imponere* (to deceive or cheat).

Meanwhile, by the time this column reaches you, Bob Toxen's long-awaited tour-de-force *Real World Linux Security—Intrusion Prevention, Detection and Recovery* (Prentice-Hall) should have been shipped, nay, airmailed, to a bookshelf or Amazon site near you. I can now reveal the identity of the Mystery Man preface hinted at in the pre-pub manuscripts I've been enjoying. No other than Eric Raymond who is much more qualified than *moi* to test and endorse Bob Toxen's efforts. Further, the book has gained a positive blurb from Steve Bourne, without whose pioneering shell where we would be today?

Check www.cavu.com/book.html for the latest price/availability news. Security is one of those ever-changing challenges. As Bob told me, he feared his book, like Tristram Shandy's memoirs, could never be completed: "It's killing me and may appear posthumously."



Stan Kelly-Boote (skb@atdial.net) has commented on the unchanging DP scene in many columns ("More than the effin' Parthenon"—Meilir Page-Jones) and books, including *The Computer Contradictionary* (MIT Press) and *UNIX Complete* (Sybex). Stan writes monthly at <http://www.sarcheck.com/> and <http://www.unixreview.com/>. Visit his home page at <http://www.feniks.com/skb/>.

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

[Advanced search](#)

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

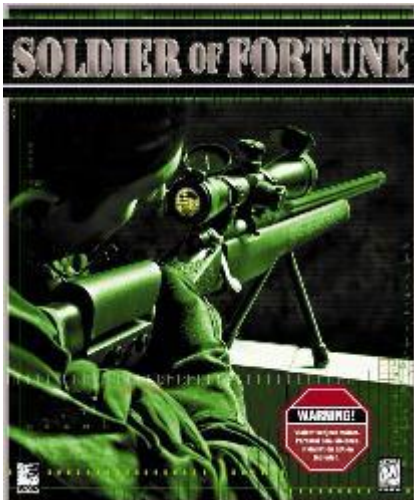
Advanced search

Soldier of Fortune for Linux

J. Neil Doane

Issue #81, January 2001

Looking at the *Soldier of Fortune* box, I was initially struck by the number of warning labels visible, each proclaiming such things as “Violent Subject Matter” and “Animated Blood and Gore”.



- Manufacturer: Raven Software
- E-Mail: sales@lokigames.com
- URL: www.lokigames.com
- Price: \$29.99 US
- Reviewer: J. Neil Doane

Looking at the *Soldier of Fortune* box, I was initially struck by the number of warning labels visible, each proclaiming such things as “Violent Subject Matter” and “Animated Blood and Gore”. And who can forget “Low-Violence Installation Option Included”. Inside, the CD-case itself was also littered with these warnings; clearly, this game has some self-esteem issues. Without reading the manual, I loaded up the software, started the game and was met with the main menu screen. Yet even *more* warnings here (I was up to about nine at this point), including a scrolling text window at the bottom of the main menu whose

message informed me of the game's "MA" rating and even how to access the parental controls in an appropriate menu. The net effect of all these warnings was perhaps the opposite of their intention; I felt a distinct urge to see what the heck these people were going on about.

I started up the first level. After a short introductory movie, I was in the thick of it, heading down a hallway to stop some terrorist bad guys and free their hostages. About three steps into the hall, the first bad guy swung out, leveling his pistol at me. I aimed the shotgun for a leg shot, hoping to be a nice guy. Boom! The shot cleanly REMOVED his leg and blew the detached portion a few feet behind him, leaving not only a bloody stump with an exposed femur and jetting arterial spray, but also producing a terrifically realistic scream of pain and agony from my would-be assassin as he slid to the floor clutching his bloody stump and literally *moaning* for a few moments before his virtual life ended. I leaned back away from the keyboard slightly with an audible, "whoa". After a short pause, a slow grin crept across my lips. "This is going to be interesting." Indeed it was.

The Story

The story for *Soldier of Fortune* is a sort of evil twist on the Rainbow Six story line. You're John Mullins, a mercenary of the highest caliber (no pun intended), who works on the darker side of the counter-terrorism world. You're on the team the good guys call when they want their dirty deeds done dirt cheap. The bad guys, a fanatical terrorist organization hell-bent on mass destruction, have seized some nukes. *Soldier of Fortune* takes you on a series of missions against this organization and its leader to recover or destroy the nukes before they can be used against the world. Each mission is worth a specified amount of money, payable upon successful completion and leads to the next mission in some way, perhaps giving clues about where the next mission should be, etc.



Figure 1. Your Partners: "Hawk" Parsons and Sam Gladstone

Unfortunately, as much as this description would seem to indicate that there is a fluid plot with specific goals, pieces to fit together and money to earn and accumulate, there really isn't. The plot itself is about as deep as a Dr. Seuss book and only randomly seems to interject coherent story line ties. The good news is that it's okay. It's a first-person shooter; we don't expect much of a plot, just a clear shot to the next bad guy and a general sense that we're getting somewhere. I admit, I kept waiting for the "hook" in the story line, the weird twist at the end that would have made it all the more interesting. When it did predictably arrive, it was a letdown, mainly because of the lack of plot development leading up to it. The aspect that you're getting paid for all this killing only enters into the game in the most tangential of ways, and no matter how much you make, you really don't get anything from the cash that you wouldn't get normally. You either win the mission and get the cash, or you play the mission until you win the mission and get the cash so you can continue to the next mission. Again, the good news is that once you start playing, the game is fun enough that you don't care.

The Game

Soldier of Fortune is, if nothing else, rather detailed in the blood-n-gore department. It uses Raven's proprietary rendering system, appropriately called GHOUL, to deliver what Raven calls "Bolt-on Gore". Using GHOUL (which also provides a number of lighting, texture, network and modeling enhancements) *Soldier of Fortune's* programmers were able to create individual characters that can react differently based on weapon impacts in any of 26 discrete "Gore

Zones". Shoot someone in the foot, they hop around for a moment yelping in pain. Shoot someone in the throat, they grasp at their throat, gurgle, and fall to the floor. They don't always die, however. I walked around one room for five minutes trying to figure out where a soft slurping sound was coming from only to find that one of my targets was on the floor, alive, yet gurgling softly through a hole in his throat. Nice touch. You can even shoot the gun from someone's hands if your aim is good enough. Parents fear not; all the really nasty stuff in *Soldier of Fortune* can be controlled via the parental controls by disabling and password-protecting such things as Damage Skins, Blood, Death Animations, Dismembered Limbs (my favorite) and so-called Adult Textures (which mainly deals with the adult language used in the dialogue subtitles.)

Though the gory details are very cool, gamers expecting *Rainbow Six*-type combat reality are likely to be disappointed with the rather *Quake*-like feel of the rest of the game. *Soldier of Fortune* actually runs on a slightly hacked version of the Quake II engine, and you can feel some of that old blockiness in the game environment. However, Raven has done some really nice work in creating interesting levels, and some especially sick and twisted levels. Most levels are littered with the typical "unlikely FPS props" such as doors that open with the push of a big red button, first-aid boxes or conveniently located ventilation system crawl spaces. The feel of the game is a bit *Duke Nukem*-like (you'll see yourself twirling a gun on your finger casually as you walk along, for instance) and even at moderate levels of difficulty, the level of combat proficiency of the AI characters you fight could probably be eclipsed by that of a two-year-old in a temper tantrum. That said, all these aspects actually work in concert to make a first-person shooter that is, quite simply, incredibly fun to play.



Figure 2: Infiltrating the Suni Secret Research Lab

Multiplayer

I found the multiplayer environment to be great entertainment. *Soldier of Fortune's* multiplayer games betray its *Quake* heritage and no cooperative multiplayer modes exist. The favorites are there, like standard *Deathmatch* and *Capture the Flag*, along with a couple new types: "Arsenal" where random weapons are assigned to you by the computer and you must score a kill with each of your weapons; "Assassin" where you must kill a specific player and defend yourself against players who are assigned to specifically kill you; and "Realistic", a multiplayer type where weapons do realistic damage to arms and legs, and medkits repair specific body parts (fixing your legs so you can run again, or your arm so you can aim again, etc.). Like *Quake*, *SOF's* multiplayer mode seems to warp space and time, shrinking what originally appears to be a long, boring evening into a short series of endless multiplayer levels and countless virtual homicides.

Installation

Loki, as usual, did a spectacular and totally seamless job porting *Soldier of Fortune* to Linux. *SOF* does require a 3-D accelerated video card and uses Mesa/OpenGL to support 3-D hardware acceleration. Loki lists the Voodoo Banshee, Voodoo2, Voodoo3, G200, G400, RIVA TNT and TNT2, and the GeForce 256 as supported chip sets. I had no real difficulty getting the software to install cleanly on a standard VA Linux Systems 6.2.3 load (a Red Hat 6.2-based system) with a Matrox G400, or on my Debian (woody) system with the XFree4 CVS tree

and a GeForce GTS. The footprint of the install is a bit hefty (about 700MB for the full install, which is the only option) and comes with both text and graphical install modes. Loki's minimums include glibc 2.1, a 2.2.x series kernel and an OSS-compliant sound card and driver. Obviously, if you want multiplayer games, you'll need a network card and either TCP/IP or IPX support (for LAN games) enabled in your kernel. Joysticks are supported. Minimum box hardware, according to Loki, is 64MB RAM, and a Pentium II processor; I had very nice performance on both a 400MHz PII and a 550MHz PIII, each with 128MB system RAM.

In Conclusion

I'd have to say that I totally dig *Soldier of Fortune*. It's fun. It's got character. It's definitely not afraid to get in your face. Most of the things I found lacking in it also are the things that make it such a great first-person shooter game. Did I mention how fun it is? If you're looking for some middle ground between *Quake* and *Rainbow Six*, this is probably your game.



J. Neil Doane (caine@valinux.com) is a professional services engineer with VA Linux Systems and an Indiana escapee. In between prolonged spasms of rabid geekness, random hardware scavenging and video gaming, he is a pilot, a guitarist and a very poor snowboarder.

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

[Advanced search](#)

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

[Advanced search](#)

The Morlock Market

Doc Searls

Issue #81, January 2001

GUI is good, but command-line computing is where it's at.

UNIX is not so much a product as it is a painstakingly compiled oral history of the hacker culture. It is our Gilgamesh epic. UNIX is known, loved and understood by so many hackers that it can be re-created from scratch whenever someone needs it. This is very difficult to understand for people who are accustomed to thinking of Oses as things that absolutely have to be created by a company and bought.

I've been reading and rereading *In the Beginning Was the Command Line*, a small book by Neal Stephenson who is better known as the best-selling author of the novels *Snow Crash* and *Cryptonomicon* (the full text is also widely available on the Web). In a general way, *IBCL* is about the real-world verities of command-line computing: its practical authenticities, its meritocratic culture, its best-tool-for-the-job approach to building and fixing stuff; and how much the GUI-using majority fails to comprehend even the existence of a better, more fundamental way to use (and not just "interact" with) computers.

But it's also about prophesy. There is an arc to Stephenson's story, one that ends where it began, with the command line. Command-line computing is not simple, he says. Nor is fixing a car or building a house. "Life is a very hard and complicated thing," he concludes. "No interface can change that; and anyone who believes otherwise is a sucker; and if you don't like having choices made for you, you should start making your own."

For the last fifteen years, the majority of the computing population has chosen to let Microsoft make their choices for them. Personally I believe Microsoft gets far too little credit for the many positive aspects of this. The fact that they turned extremely complicated processes and functions into moderately complicated but extremely appealing products is a marketing triumph of the highest order. Today it is no more possible to do business in the world without

touching Microsoft products than it is to find transportation without internal combustion engines.

Worse, these products' frequent failures are legitimized by ubiquitous acquiescence. Jeff Rankin says, "Imagine if every Thursday your shoes exploded if you tied them the usual way. This happens to us all the time with computers, and nobody thinks of complaining."

Stephenson isolates at least two Faustian reasons for this. One is humanity's fondness for mediated experiences. Witness the success of Disney, which "does mediated experience better than anyone," Stephenson writes. "If they understand what OSeS were, and why people use them, they could crush Microsoft in a year or two." The other is that "we are way too busy, nowadays, to comprehend everything in detail. And it is better to comprehend it dimly, through an interface, than not at all."

The key word is "everything". Personal computing was born with ambitions that far exceeded its abilities. Because it *could* do just about anything, it *should* do just about anything. And, amazingly, there was more than sufficient demand for enough of "just about anything" to justify and attract venture funding for software start-ups by the multitude. But in the long run (which, again, hasn't really been very long), only one company seemed to understand exactly how much of *everything* could practically be handled by a PC, and how to minimize the inherent complications for the largest percentage of *everybody*. However awful Microsoft may have been in other ways, this comprehension alone is an achievement of Roman dimensions.

The cultural result is what Stephenson calls "a two-tiered system, like the Morlocks and the Eloi in H. G. Wells' *The Time Machine*, except that it has been turned upside down". Here is his explanation:

In *The Time Machine*, the Eloi were an effete upper class, supported by lots of subterranean Morlocks who kept the technological wheels turning. But in our world it's the other way round. The Morlocks are in the minority, and they are running the show, because they understand how everything works. The much more numerous Eloi learn everything they know from being steeped from birth in electronic media directed and controlled by book-reading Morlocks. So many ignorant people could be dangerous if they got pointed in the wrong direction, and so we've evolved a popular culture that is almost unbelievably infectious and neuters every person who gets infected by it, by rendering them unwilling to make judgments and incapable of taking stands.

Morlocks, who have the energy and intelligence to comprehend details, go out and master complex subjects and produce Disney-like sensorial interfaces so that Eloi can get the gist without having to strain their minds or endure boredom.

To be fair, Stephenson goes on to credit the positive social effects of mediated experience:

The spectra of a policy controlled by the fads and whims of voters who actually believe that there are significant differences between Bud Lite and Miller Lite, and who think that professional wrestling is for real, is naturally alarming to people who don't. But then countries controlled via the command-line interface, as it were, by double-domed intellectuals, be they religious or secular, are generally miserable places to live.

The cultural distinctions are interesting but frankly not important. The real and important division is between makers and users. Let's divide the computing world into three classes and look at how the Morlocks and the Eloi sort out:

The desktop is Eloi territory. Apple conceived it, Microsoft owns it, and most of us populate it. The server is a bit of a mix. It's a Morlock business, but with a lot of Eloi sensibilities. This is why Linux and Windows NT/2K are both growing in absolute numbers and market share (and I want to be sure not to insult the countless Morlocks who hack righteously both on and with various Windows products). The embedded world is all Morlock. It always was, and always will be.

The notion of a device that does everything is ludicrous in the embedded world, which is comprised of nothing but specialties. The embedded world also doesn't need fancy metaphors because nobody wants a button or a dial to do "whatever", depending on which application is running. There is no "whatever" in the embedded world. If you're dialing a radio or regulating a valve, you're doing it on a device intended to do just that and not much more.

It turns out that Linux, by virtue of its small size, modular form, familiarity and open-source code, is ideal for an infinitude of single purposes. It's also exceedingly practical. This is why the Morlocks will soon be hacking away at everything that can conceivably benefit from Net-connected embedded intelligence. Since this includes a vast amount of stuff, we can expect the Morlock population to quickly grow in number, diversity and power. The result will be a revolution far more profound and important than personal computing.

For the suits among us, the most important question might be, *How long before Linux, which manifests as practical specialization, makes personal computing as we know it obsolete?* In other words, when will it be easier and faster to hack together (or buy, or both) a point-of-sale system that runs on Linux, rather than cope with a third-party package that has to run on crusty old Windows98? Or an accounting system that does accounting and little more, but connects to the rest of the world over TCP/IP and runs on reliable generic hardware?

Let's put it another way. *How long before nobody gets fired for specifying Linux because too many of the suits are Morlocks?*

Here's a good place to start: it's already that way at IBM.



Doc Searls (info@linuxjournal.com) is senior editor of *Linux Journal* and coauthor of *The Cluetrain Manifesto*.

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

[Advanced search](#)

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

Advanced search

TuxTops Obsidian N30W

Jon Valesh

Issue #81, January 2001

Every TuxTops laptop comes with Linux pre-installed.

- Manufacturer: Tuxtops Inc.
- E-Mail: info@tuxtops.com
- URL: <http://www.Tuxtops.com/>
- Reviewer: Jon Valesh



Buying a laptop computer teaches you all about confusion and sometimes about yourself too. Nothing brings out anal-retentive dithering like considering spending thousands of dollars on something you can tuck under your arm. Nothing sparks confusion like going through dozens of nearly identical but critically different specifications, searching for some clue about how you should spend your money.

It seems simple enough when you start: maximize the features, minimize the price. Simple does not describe the details, though. The details can drive you insane. Screen size, CPU type, DVD/CD/CD-RW drives, hard drive capacity,

battery life, weight, mouse substitute, expansion ports, video capabilities—just describing it is enough to make a person hyperventilate. Not only is it an immense list, every item represents a trade-off. Want the best battery life? Be prepared to carry a couple of extra pounds or have a weak processor and few features. Want the biggest screen? It won't look as good. The fastest CPU? The batteries won't last as long. Lots of expandability? Everything gets bigger.

And the safe shortcuts that people fall back on aren't always reliable. Buying by brand is a good way to donate an extra thousand dollars to a big company for the exact same product. It is not unusual to find three laptops, each supposedly from different makers, which are exactly the same unit: the same OEM, the same expandability and the same hardware capabilities. But not the same price—far from it. Just to confuse things, sometimes supposedly identical laptops sold under different brand names seem the same, but do not have exactly the same features. The display available from one vendor may not be sold by any of the others, though all are the same core unit.

Adding Linux compatibility to your requirements list only makes matters worse. Virtually every laptop has *something* that Linux can't quite take advantage of. Maybe the power-saving suspend feature won't actually suspend; maybe the sound is difficult to get working; maybe everything will work fine once Linux is installed, but you can't install unless you have a network because the CD-ROM isn't recognized by any of the standard Linux boot floppies.

Whatever mix of requirements you have, finding a laptop that does what you need can involve a lot of Net time, a lot of searching and a lot of faith in the manufacturers. Even when your requirements seem simple, it is seductively easy to start out looking for a \$1,200 economy laptop and end up buying a \$3,400 full-featured desktop replacement. Every step seems so reasonable that you hardly even realize what you are doing until the bill comes. Car dealerships could learn a lot from laptop manufacturers about up-selling their products.

But, after your nervous breakdown and before the credit card bill arrives, there is something really wonderful about a great laptop. It may have cost more than a nice used car, but you can't take a car into a restaurant either. Knowing that you are carrying around enough computing power to cause a gadget freak from ten years ago to pass out is a pleasure that extends beyond pure geekdom: anyone can understand it.

The TuxTops Obsidian is exactly that sort of machine. As close to a no-compromise laptop as you can buy, it combines battery life, performance, display and video capabilities and even ergonomics in a package that will make most people happy—if anything will. Most importantly, every TuxTops laptop

comes with Linux pre-installed. Yeah, that other desktop OS is available, but Linux is the main deal here.

So, What's It Got?

The TuxTops Obsidian is a top-of-the-line laptop, and the test unit was configured to prove it. Start with a 650/500MHz SpeedStep Pentium III, 256 Megabytes of RAM, an 18 gigabyte hard drive, 8MB ATI Rage mobility LT video chip set, 6 speed DVD drive, a 15-inch 1024x768 display and all of the usual features like sound, dual PC-CARD/Cardbus slots, a floppy drive, a built-in Lucent Winmodem, IrDA, serial and parallel ports, a port replicator connector for desktop use and more, most of which are ready to use the first time the system boots into Linux.

In laptop jargon the TuxTops Premium is a three-spindle design, meaning that the floppy drive, hard drive and CD/DVD drive can all be installed in the laptop's case and used at the same time. The CD/DVD is actually in a quick-release media bay that can accept a CD/DVD ROM, ZIP, CD-RW or even a second battery.

Physically, the TuxTops Premium is about as no-nonsense and as tough as you can get without resorting to titanium. The case is a tough black plastic that seems immune to casual scratching from watchbands, keys and just about anything else. The age-old problem of break-off dust covers that weren't meant to break off has been solved by leaving the covers out entirely. The only part that really seems vulnerable is the large display. It twists noticeably if adjusted by holding only one corner. If you have a diskette in the floppy drive, the eject button extends rather far, preventing you from storing a disk in the drive while the computer is in its carrying case.

The keyboard has an above average feel, but in my opinion the key layout is a bit strained in places. For example, the home, end, page up and page down keys are at the top of the keyboard and the regular cursor controls are at the bottom. I prefer laptops that use the FN key to merge all of the cursor controls into the same four keys. The FN key is in general rather under-used, with only five or six functions, mostly related to controlling the hardware. Of the standard keyboard keys, only SysRq, Scroll Lock and Break require the use of the FN key.

The mouse is a two-button touch pad, which can be a critical issue for some users. Emotions run strong. People either love them or hate them, and arguments about the most appropriate laptop pointing device have been known to degenerate into fist fights. You should spend a few minutes using a laptop with a touch pad before you make up your mind, but if your thumbs tend to drift around below the space bar, you may find your mouse pointer

bouncing around when you least expect it. When used in X, the two buttons must be chorded—clicked at the same time—to emulate the third mouse button. A minor inconvenience for normal use, but tedious with software that makes extensive use of the third button.

There is also a selection of connectors and buttons that aren't used in Linux, such as the single USB connector, the push-button style external volume control and, depending on your kernel version, the Lucent WinModem.

Reliability

Having a laptop fail is not only more expensive than the same failure in a desktop computer, but more likely: you carry laptops around and what gets carried gets dropped. Or, if not dropped, mistreated in other ways. A laptop lives a dangerous life.

A real laptop test would involve a few well-staged drops from tables or small office buildings. A real test would...but I'm a coward. Cheap, too. I didn't do anything to intentionally damage the TuxTops laptop, but I haven't been gentle either.

What I have done is carried it almost everywhere for the last month or so. It has traveled over 1,700 miles (I counted) over six-lane highways and dirt roads, shared table space with dozens of hamburgers, been lugged in and out of office buildings, houses, shopping malls, public parks and anywhere else I could safely go. It has been opened and closed, demonstrated to strangers, shoved haphazardly into its carry bag to allow quick escapes from coworkers and other undesirable associates, been tossed into trunks, loaned to friends and generally abused as much as I could abuse it without actually, officially, trying to harm anything.

I have given it the workout that a truly laptop-enchanted geek would give their computer if it wasn't going to be their computer forever and if they didn't have to pay for repairs. I've done my best to put a year of use into a month of time—and the result has been enlightening.

The worst problem is that the display is difficult to clean. I have learned that laptop displays and fast food don't mix, which says more about the American diet than the reliability of this laptop, though.

There has been an assortment of little problems, some trivial, some preventable and some troubling, but none critical. Perhaps the most trivially bothersome problem has been the little rubber feet. They come off if you try to slide the computer across a desk or tabletop. So far, they have always reattached themselves when pushed back into place, but it would be easy to

lose them. Several of the screws securing the various access covers on the bottom of the case showed a tendency to come loose, though that may have been caused by my surreptitious opening of each cover after receiving the laptop. The most troubling problem isn't really a problem, but it could become one: there seems to be a loose screw or another small part rattling around in the case, near the floppy drive. Whatever is rattling around hasn't caused any damage yet, but the laptop wasn't rattling when I received it.

Is It Cool Enough?

Cramming a 650MHz or faster CPU into a tiny case without paying careful attention to getting rid of the heat modern CPUs produce is a recipe for melted plastic. Modern laptops get hot—sometimes very hot. The TuxTops Obsidian uses a flow-through heat sink with a thermostat controlled fan that draws air from the side and pushes it out the back of the case when the CPU temperature climbs. The fan is very small, and in a quiet office or home environment the sound it produces can be very noticeable. In a restaurant or car you'll probably never realize it has turned on, though. Perhaps literally—the first person I loaned the TuxTops to, returned it after less than 20 minutes of use, citing a fear that the unit would melt if left on. It does get hot (hot enough to keep it off your lap, unless you have thick pants) but not hot enough to melt. This is normal. The designers chose to use the bottom of the case for heat dissipation, helping them reduce their dependence on the power-wasting fan.

Running the computer with a blocked air vent is definitely not advised, though. After an hour or so the smell of hot plastic will probably give you a headache.

Battery Life

The single Lithium Ion battery provided about three hours of normal use when using the SpeedStep feature to reduce the CPU speed to 500MHz and using the automatic power management in the BIOS. The life was a little longer in some cases, with the longest being about three and a half hours. It was significantly shorter with heavy disc use or when the CPU was turned up to full speed.

For extended use, you can easily install a second battery by removing the DVD/CD-ROM drive and hot swapping the batteries to keep the computer running as long as you have charged batteries. The system will automatically fully deplete the first battery before starting on the second, so you can easily manage your battery use.

The batteries are, at \$100 to \$200 each (depending on when and where you buy them), priced similarly to other batteries of the type. In other words, they are very expensive. The extra price is well worth it for the long run times these batteries provide. Each battery has its own microprocessor to monitor the

charge status and control the charging appropriately for the particular battery. There is a built-in charge indicator that will, at the touch of a button, light an LED bar graph to indicate the charge state of a battery that is not installed in the laptop, so you can keep track of which batteries you've used and which are ready for use.

What about Linux?

Linux support is where TuxTops really shines. The folks at TuxTops are real-life Linux users, not just suits who saw some advantage to offering "Linux for the weirdos". The folks at TuxTops know and use Linux, and they know and use their own products, so when you ask a support question they can answer it from personal experience. The test laptop came with an unofficial Red Hat 6.2 release pre-installed and configured to take advantage of the Laptop's hardware. All of the hardware drivers are pre-installed, as is the XFree86 configuration file for the LCD display. The first time you boot your TuxTops Obsidian, you are asked to supply some basic information to configure features such as networking. After that, the system will always boot into an X11-based login screen.

Also included was a bootable CD-ROM with Red Hat and several laptop-specific RPM packages ready for installation. Reinstalling Linux is as easy as booting from the CD and acknowledging the repeated warnings that you will lose all Linux files on the hard drive. Unfortunately, there is no way to specify how the hard drive will be partitioned, or which partitions will be formatted, so you will always have a large /home partition with new files on it. This reduces the value of having a /home partition at all.

The biggest problem with the Red Hat installation was actually the lack of choices when installing it. TuxTops installs the software you need, and the result is a laptop where everything that can work, works, but a lot of the choices have been taken away. The default desktop is GNOME, and while KDE is installed there is no option to make it the default-user environment. Likewise, there are very few choices about how the rest of the system is installed and what software is used.

The folks at TuxTops report that by the time this article is in print they will be offering a wide range of Linux distributions pre-installed on the Obsidian and all of their laptops. I didn't have a chance to try out any of the other distributions, but if they are as well configured as the Red Hat installation, TuxTops will make a lot of people who want or need other distributions happy.

And the Other Operating System?

The review laptop came in a dual boot configuration with Red Hat and Microsoft Windows98. The Windows installation was about as bare bones as you will find in a computer. The operating system was installed, but there was no user software—no office suite, no DVD player software, nothing but Windows. This makes sense from a company that really isn't trying to sell Windows at all, but you should be aware that if you buy a dual boot system, you will need to add your own software to make the Windows installation useful.

Summary

The TuxTops Obsidian provides top-of-the-line performance, great expandability and great support at a price that is a lot lower than nearly identical hardware from other vendors. If you are looking for a desktop replacement system or a highpowered laptop, it is hard to go wrong with the TuxTops Obsidian.

The Good and The Bad

Born at the beginning of the microcomputer age, **Jon Valesh** (jon@valesh.com) has pushed and been pushed by computers his entire life. Having run the gamut from games programmer to ISP system/network administrator, he now occupies himself by providing technical assistance to ISPs and small businesses whenever his day job doesn't get in the way.

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

[Advanced search](#)

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

Advanced search

SuSE Linux 7.0

Stew Benedict

Issue #81, January 2001

If you like to measure your Linux distributions by the pound, SuSE Linux Professional 7.0 wins hands down.



- Manufacturer: SuSE Inc.
- URL: <http://www.suse.com/>
- Price: \$69.99 US
- Reviewer: Stew Benedict

If you like to measure your Linux distributions by the pound, SuSE Linux Professional 7.0 wins hands down. With 1,500 applications on 6 CDs and a DVD, you'll be hard pressed to come up with something that is not included.

In addition to the above media, SuSE 7.0 comes with two floppies, a boot disk and a modules disk and four books, *Quick Install Manual*; *The Handbook*, a technical guide; *Configuration*, which covers KDE, hardware installations and general Linux command-line usage; and *Applications* which covers StarOffice, Netscape, Acrobat Reader, the the GIMP, Sane and other multimedia applications.



Figure 1. The SuSE Linux 7.0 Package

The SuSE Linux 7.0 Package is the “other” OS for my CAD work. I had a 6GB drive available, so I pulled the other drive and dropped this one in. This is just being overly cautious, I never know when I'll get a call and need to run that “other” OS, and I need to be able to get back up ASAP. I did another install before this one, and the installer does recognize and offer to setup LILO for other OSes. The machine is a Pentium 166 with 80MB RAM. I went into the BIOS set up, re-scanned for the new HD and set the boot options to boot from CD.

This brought me to a text-based screen that said “Have a Lot of Fun!” and then proceeded into a normal kernel boot, scanning for devices, etc. One addition I noticed was the kernel scanning for braille devices—a nice touch. In fact, the box says SuSE Linux is entirely suitable for use by the blind. I also noticed that software RAID support is included in the provided kernel. I recently implemented this at a client's site, and it has worked out quite nicely, with two drives in a RAID 1 array for a redundant, mirrored system.

From the initial boot, the screen went blank, the speaker emitted a beep and that was it! At this point, according to the install, I should have been in YaST2, the SuSE graphical installer. Apparently I have issues with my video card, an ATI Mach64-based card. The notes in the CD folder mention typing “manual” at the LILO prompt for expert installation with YaST1, so I reboot and try this route, and the install goes into YaST1. This time I get to a “dialog”, text-based install that asks my preferred language and proceeds to the option of a text-based YaST (see Figure 1), or graphical YaST2 (see Figure 2). I decide to live dangerously and try YaST2. This time I do get to the graphical interface, a little grainy at 800x600, but functional. So far, no mouse, but I can tab through the options, and it gets me to a screen where I can select my mouse type and port. Now we are off and running. Next we get to the keyboard layout and time zone, after which I am asked whether I want to upgrade or do a fresh install. I choose the fresh install. Next we have a choice of installs:

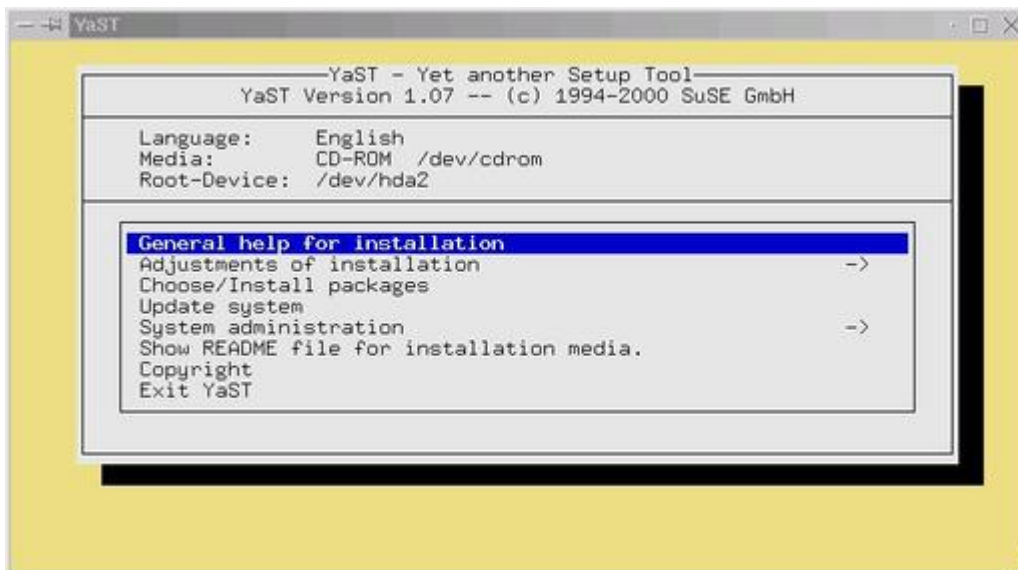


Figure 1. YaST

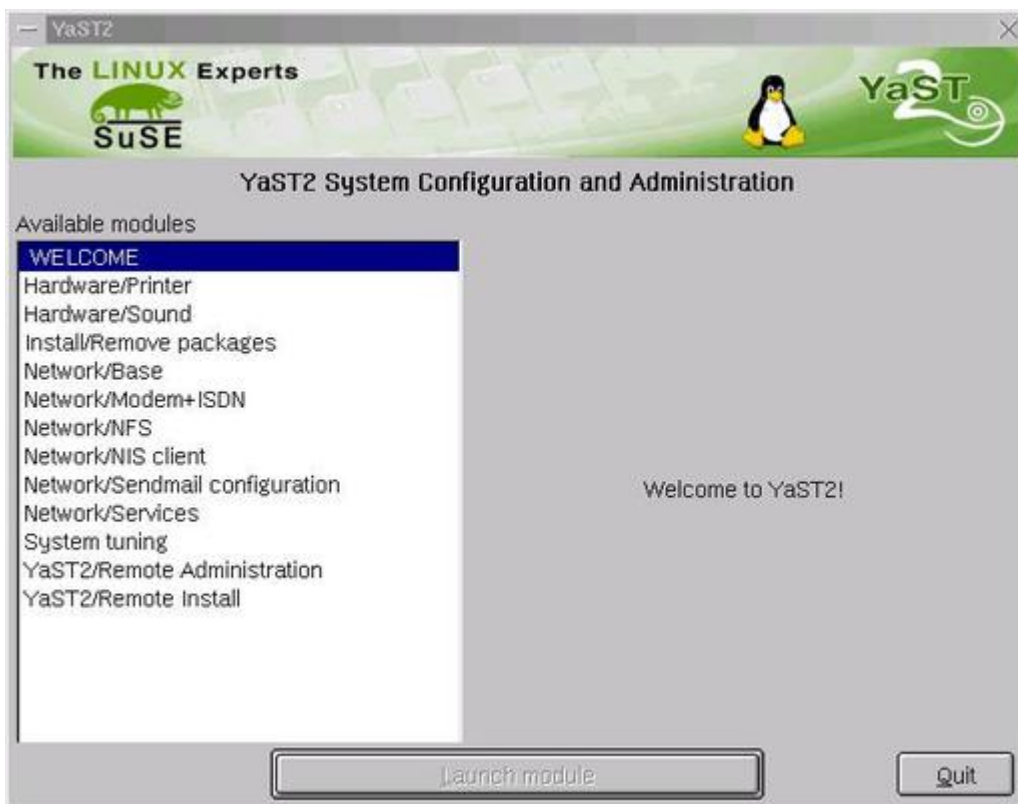


Figure 2. YaST2

- Almost Everything
- Minimal
- Default
- Default with Office
- Detailed Selection.

I choose Default with Office so I can appraise what gets installed by default.

Next I'm asked where I would like LILO to be installed and go with the suggested `/dev/hda...`. Then I personalize the installation with my first name, last name, login and password, and lastly, the root password I would like to use.

Partitioning

The default partitioning scheme sets up just three partitions: `/boot`, `/` and swap. I back up and opt for custom partitioning, which allows me to choose between ext2 and ReiserFS. I set up `/` and `/boot`, `/usr` and `/home`. The install gives you an option at this point to save your settings to floppy so, in case there is a problem, you can get back into where you left off.

While we're on partitioning, I should mention a little about ReiserFS. This is a relatively new journaling file system for Linux that boasts certain improvements over the standard ext2 file system. Journaling, if you're not familiar with database systems, implies that each file system transaction is written to a journal or log. It's possible to replay this journal in the event of a catastrophe and recover the lost transactions. The journal is flushed periodically once it is certain the transactions have actually taken place.

Here's the freshmeat.net entry on reiserfs:

reiserfs is a revolutionary new approach to file system design which stores not just file names but the files themselves in a B*-tree. It is a generation ahead of alternatives that use older, plain B-tree technology and that cannot store the files themselves in the tree.

Since I have some extra space on this drive, and don't quite know what to expect with ReiserFS at this point, I set up duplicate `/home` and `/usr` partitions, one with ext2 and one with ReiserFS. The chart below shows the layout I came up with:

`/ 1GB ext2 /boot 7.8MB ext2/usr 1.5GB ReiserFS/usr2 1.5GB ext2/home 900MB ReiserFS/home2 900MB ext2swap 160MB`

I'll cover more on ReiserFS later in the article, once we have things installed and set up.

Package Installation

The partitions are set up, and we move to the installation of 341 packages. As the install progresses, you get a typical status screen displaying the package name being installed, a short description, and an overall progress update and percentage of disk space consumed. The installation I selected uses a little over 1GB of space.

At 242 packages, I am informed that the LILO boot sector has been set up, and I can restore the old one with the command:

```
lilo -u /dev/hda
```

Then the installer starts the new system, and a text screen tells me the base system has been successfully installed. The graphical YaST2 continues, asking for CD2, then resumes installing packages. Another pause at package 326, and I'm asked for CD3. Another pause at 331 for CD4.

X Setup

Now we move on to X set up. I am asked for my monitor, a Mag MX17, and then the video card, which it correctly identifies as an ATI and sets for 1240x1168@16bpp. I try the test at this resolution, my color palette shifts to an unusual look and the system appears to lock up. I can't switch to a different VT, nor does CTL-ALT-Backspace or CTL-ALT-DEL have any effect, so I resort to the reset button. The system boots into SuSE and runs checks on the file systems since it wasn't shut down properly, then gets me back into YaST2 at the monitor configuration. This was nice. Even though I hit a major problem, the install recovered and resumed where I left off. This time I try a more conservative setting, 640x480@8bpp—and get a black screen and another lockup. Hmm. Seems the ATI driver has a problem with my card. There is no option to override with a generic SVGA card, so I opt for no X11 and move on. The install does tell me I can set X up later with SaX.

Peripherals/Networking

From here, YaST2 moves on to setting up the printers, sound, Internet and network. I do not have a modem on this machine, and YaST2 tells me it has not detected a modem or ISDN adapter. It does recognize my Creative sound card and my 3Com NIC. I go ahead and set up my IP address with the default gateway through my file server and a host name with DNS pointing to my ISP's name servers. The install checks to certify that I have entered valid IP addresses.

YaST2 tells me the NIC was configured correctly and networking is set up, which I verify by pinging the box from my file server. Looks good!

The sound card setup allows me to set the volume and test, and it plays a nice little orchestral piece—good again! Later on I look at the loaded modules, and it looks like sound is handled by ALSA.

For printing, I have an HP print server on the LAN with a NEC Silentwriter and an HP Deskjet 693C. The installer scanned the local LAN for hosts and failed to

find anything, probably because I'm not running DNS on the server. Since I have a custom lpr script on the server that powers up my printers via an X10 control module when I need to print, I route the print jobs to the file server, using its IP address, and it takes care of the filtering too. One minor annoyance here is the first network printer is called "remote" with no opportunity to override that name except on the second printer. I would rather have preferred setting it to something more descriptive.

That was it on this screen, so I moved on to "Finish Installation", and was told I could log in as "stew". The manual reminds you to change your BIOS settings back to boot from hard drive and/or floppy.

First Login

I log in, do a `ps -ax` to see what is running—looks normal enough, `httpd`, `lpd` and several copies of `nscd`, which the man pages tell me is a name server caching daemon—okay. `lynx localhost` gets me to some local SuSE pages—good. My server is on line, so I try `lynx` on some remote web pages, and they come up too. Looks like networking is working fine.

Back to X Setup

I take a quick visit to the SuSE web site and search their support database to see if I see anything on my X issues, but the info seems to refer only to SuSE6.4 and older versions, so I forge ahead on my own.

To see if I could pick up anything on my X issues I do an

```
rpm -q -a | grep mach
```

from the command line. I see the following:

```
xmach64-3.3.6-44
```

This explains my problems since YaST2 was trying to use the XFree4.0 server. Doing a similar search on `xf`

```
rpm -q -a | grep xf86  
xf86-4.0-55  
xf86_3x-3.3.6-44
```

It looks like sticking with XFree3.3.6 would be the wise thing to do. I log out and re-log in as root to finish setting things up.

I was able to run SaX from the command line and correctly configure XFree3.3.6 by manually choosing my ATI video card, an Xpression card. Then I was able to fine tune the settings and get it set up for 1024x768@16bpp. The card is an older card with only 2MB VRAM, so this is about as good as it gets. Running

startx, I am in the KDE desktop. SaX uses XFree 3.3.6 vs. SaX2, which uses XFree4.0.

Running X

Okay, now we can run **startx** and get into X, which defaults to the KDE desktop environment. I run the KDE control panel and try some of the settings. System sounds are disabled by default, and I notice that none of the default KDE sounds are associated with their actions. I set up the associations and enable sound, and they work as expected. I also tried an audio CD with KSCD, and it plays as expected, as did an MP3 I grabbed off my file server.

The rest of the KDE settings for colors, desktop, etc., are the same as those with which I'm familiar in other distributions. This is KDE 1.1.2, not the version 2 that everyone is talking about, although the beta comes on the SuSE 7.0 CDs.

If you prefer different Window managers or desktop environments, those are included too. I did not see the switchdesk application, common on Red Hat-based systems. When you have X working though, you can enable the graphical login kdm via YaST. Once this is done you have a choice of KDE, Windowmaker, twm or fwm2. An interesting tool is DyDe (see Figure 3), a SuSE dynamic desktop configurator, that allows you to mix-and-match window managers and desktop environments on the fly. Once I went back into YaST2 and installed them, GNOME, Enlightenment, Sawmill and Xfce were also available.

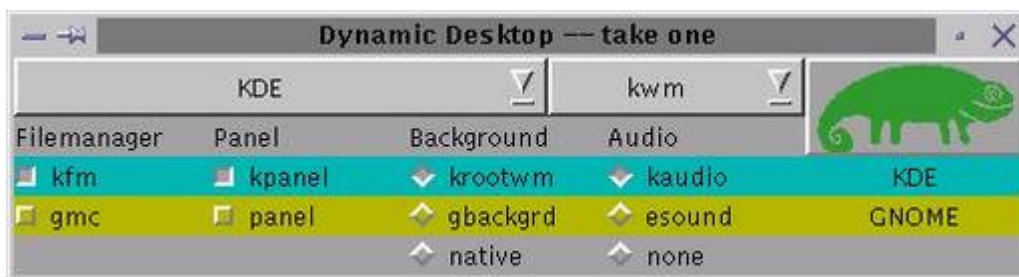


Figure 3. Dynamic Desktop Configurator

I was pleasantly surprised to find a new version of the GIMP, version 1.2 prerelease when I went to do the screenshots for the article. The GIMP is the favorite graphics tool of most Linux users, and it looks like the new version adds some nice features. Xsane was also installed by default and works nicely with my HP ScanJetIICX.

The Kernel

The kernel version installed is 2.2.16, although the box claims 2.2.17-pre. Looking on the CD, it looks like source is there for version 2.2.16 as well as an assortment of precompiled modules in **/lib/modules/2.2.16**.

User Login

Logging in as a normal user, the KDE desktop is set up with icons for Netscape, StarOffice and icons for both my CD and CDR, my Zip drive and Bernoulli drive. You are also greeted with a SuSE welcome screen (see Figure 4). Clicking the StarOffice icon launches the StarOffice 5.2 Workstation installation, which lets you do an individual install for each user. This only requires 1.6MB of space per user in their home directories. StarOffice still uses their Star Desktop work space, but I understand that they will be breaking the applications up in the open-source version to come. With Netscape 4.73 up and running, I initiated a call to the Internet from my server and successfully navigated to the SuSE web site, so my gateway setup is working. The local Apache server launches a cute page with links to local documentation, as well as SuSE's site (see Figure 5).

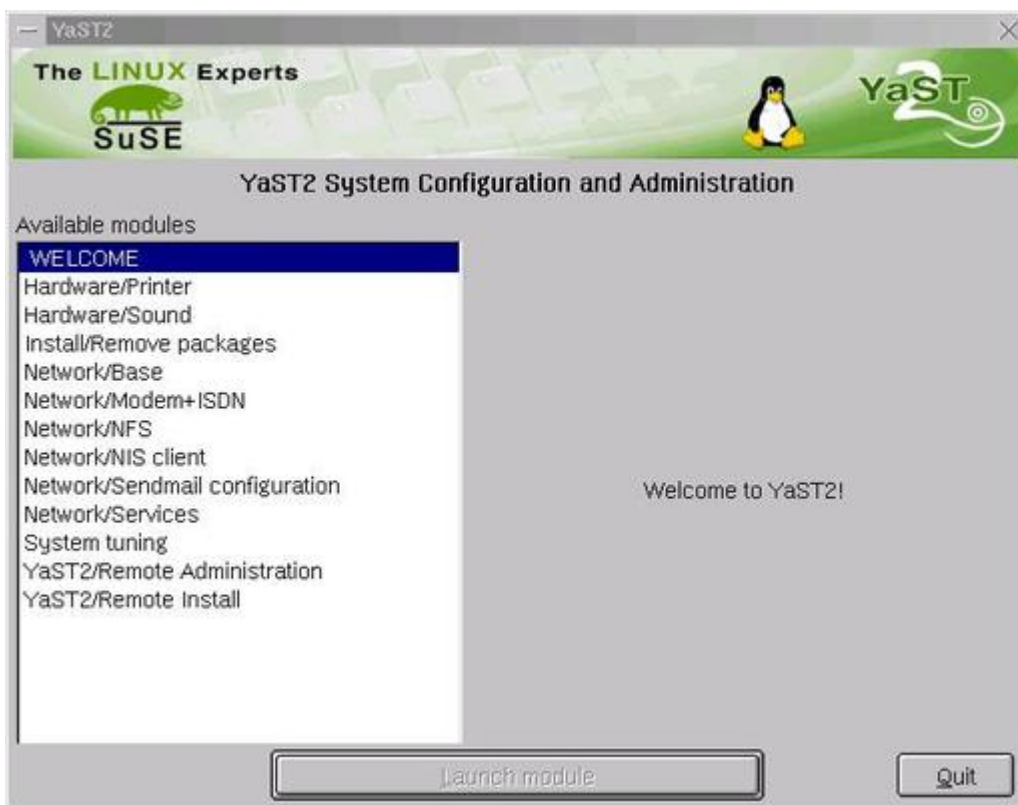


Figure 4. SuSE Welcome Screen



Figure 5. SuSE Links Page

Even after all these years, pine is still my e-mail reader of choice, and there it was, like an old friend, along with fetchmail. If you go for the GUI e-mail programs, kmail and Netscape messenger are installed by default with many more on the CDs.

There's also an alternate file manager for KDE—kruiser, which is an MS Explorer-style file browser.

On the commercial applications side, there are an assortment of free and demo versions including VMware, ADABAS D Personal Edition, Arkeia Backup, Lutris Enhydra, Hummingbird Exceed and Via Voice.

I won't go into all of the packages installed or on the CD, or this would turn into a SuSE 7.0 supplement. Suffice it to say with the wealth of applications provided, I don't think you will be needing to go to freshmeat or rpmfind for a while.

ReiserFS Tests

Since the journaling aspect of ReiserFS seems to be what everyone talks about, and this system is essentially virginal, I decided to do a completely unscientific test. I backed out of X, reverting to the graphical kdm login, then switched to VT1, logging in as root, typed "sync" to sync the file system and did the unthinkable—hit the power switch. As you might expect, when I booted back up, the kernel notes that the file system was not closed out properly and proceeds to run fsck on the ext2 partitions. For the `/usr2` and `/home2`, at 1.5GB and 900MB, this takes 20-40 seconds each and recovers without any serious issues. Remember, I did do a sync, which flushes the disk cache buffers. For /

`usr` and `/home`, which are ReiserFS, the kernel runs `reiserfsck`, and it completes two to seven journaled entries in one second and moves on—nice! Just for laughs, I repeated this test a couple of times with and without syncing, with similar results. Don't use this as your first line of defense for catastrophic failure, but it does look like a good way to get up and running quickly in case of such an emergency.

The messages I read on the Net talked of speed issues in using ReiserFS instead of ext2. To get a look at this, I downloaded `bonnie` from `freshmeat.net` and ran a couple of tests. In both cases, I was the only one logged into the system via Telnet with `kdm` running but no X session. All the rest of the system `dæmons` were those running by default.

Listing 1

On the single 200M file there did not seem to be much difference. On the 30 files tested, it looks like ReiserFS does have a bit of a cost in terms of speed, looking at the `/sec` readings. Again, this is not exactly a controlled, scientific test, although I did run the tests a second time with similar results. I also ran them as root, so it did not seem to be a permissions issue. In actual use, I did not notice any performance hitch in running applications. Whether you want to go for the speed of ext2 or the safety of journaling is up to you. I'll leave ReiserFS running on this system for a while.

Conclusion

In summary, it looks like SuSE 7.0 Professional has most of what Linux users want in a distribution and then some. The “Default with Office” had almost everything I would want in a desktop system. All of my hardware was detected and set up except my IDE-CDR, which would require a kernel recompile to burn CDs with it. I did have the initial problem with my ATI video card and YaST2, but the install had a workaround for this. I did not call SuSE support, but the package includes 90 days of installation support via telephone, e-mail and fax. SuSE's web-based support database is pretty comprehensive as well, although apparently 7.0 is still too new to have many entries. I recently did a 6.4 install on my iMac, and several of the issues I had there were covered in the database. If you are accustomed to a Red Hat-based distribution, some of the configuration tools and file locations, as well as the init scripts are a little different, but not a big issue in my book. For the \$69.95 US they are asking at SuSE Shop, it's a steal compared to some other OSes.

Stew Benedict is a systems administrator for an automotive manufacturer in Cleveland, Ohio. He is also a freelance consultant, running AYS Enterprises, specializing in printed circuit design, database solutions and utilizing Linux as a low-cost alternative to commercial operating systems and software. He has

been using and promoting Linux since about 1994. When not basking in the glow of a CRT, Stew enjoys time with his wife, daughter and two dogs at his future (not too much longer!) retirement home overlooking Norris Lake in the foothills of the Smokies in Tennessee.

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

[Advanced search](#)

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

[Advanced search](#)

StarOffice 5.2

Stephanie Black

Issue #81, January 2001

It installs without a hitch, and it no longer requires every iota of available memory to run, unlike its predecessors.

- Manufacturer: Sun Microsystems
- E-Mail: so_inquiries@sun.com
- URL: www.sun.com/staroffice/get.html
- Price: Free download (shipping and handling for media)
- Reviewer: Stephanie Black

The need for business applications that run on Linux is well documented. We all want something that will further diminish the hold of the Redmond Contingent on the corporate market. We want to be able to cheer on anyone bold enough, brash enough and bright enough to create a successful rival to Microsoft Office 2000. "Successful" is the key word here.

This certainly is an office suite. The manual says you can do everything administrative from "one place". It's described as "intuitive". The first claim assumes that the user has no temporal requirements with respect to those administrative tasks: it can take a while just to find the required administrative task. The second claim assumes that the user has been doing this sort of work for far too long, only under one other operating system. "Star" quality is elusive.

Granted, it doesn't cost a few hundred units of your local currency, even if you want the media. It installs without a hitch. It no longer requires every iota of available memory to run, unlike its predecessors. It's not a "special" or "millennium" edition. It's available for several different platforms. It doesn't crash when you open it. Technically, all the pieces are there in nearly the exact order as described in the manual. It apparently works as a word processor, presentation tool and Swiss Army Knife clone. And, I have to admit, the technical support available for it really is superb.

The problem is, someone decided StarOffice should follow St. Paul's dictates, to wit: to be all things to all people. StarOffice is not, and will never be, all things to all people. If it manages to be a simple suite of office software that is clean, original, functional and comprehensible, it will be enough for thousands. Especially if running it on 64MB of RAM makes for a quick initialization, as opposed to one during which you can talk with your lawyer, editor, mother-in-law and local bill collectors and still have time to get a cup of Starbucks Special before the software is ready to actually use.

Hint: there's a Start button at the bottom left of the application window, and it's labeled that way. The butterfly pixmap makes no difference to the functioning of the software. Really.

Installation Notes

You will not necessarily require a whole Gig of either hard drive space or RAM to install StarOffice. Both, admittedly, might be helpful if you're going to use really large databases, but they're not necessary. I chose a custom installation of about 250MB running on SuSE 6.4. You can choose parts of packages, if you don't want the full allotment of graphics, templates, effects and backgrounds, but wish for only a selection of those gallery items that suit your purposes. This applies to other components of StarOffice as well. For this review, Draw, Image, Basic and Calc were omitted.

Note: StarOffice includes a kind of BASIC for macro and script creation. This would have usable applications under Windows but under Linux seems rather misplaced, given that C is the mother tongue of Penguinese and Bash.

StarOffice Desktop

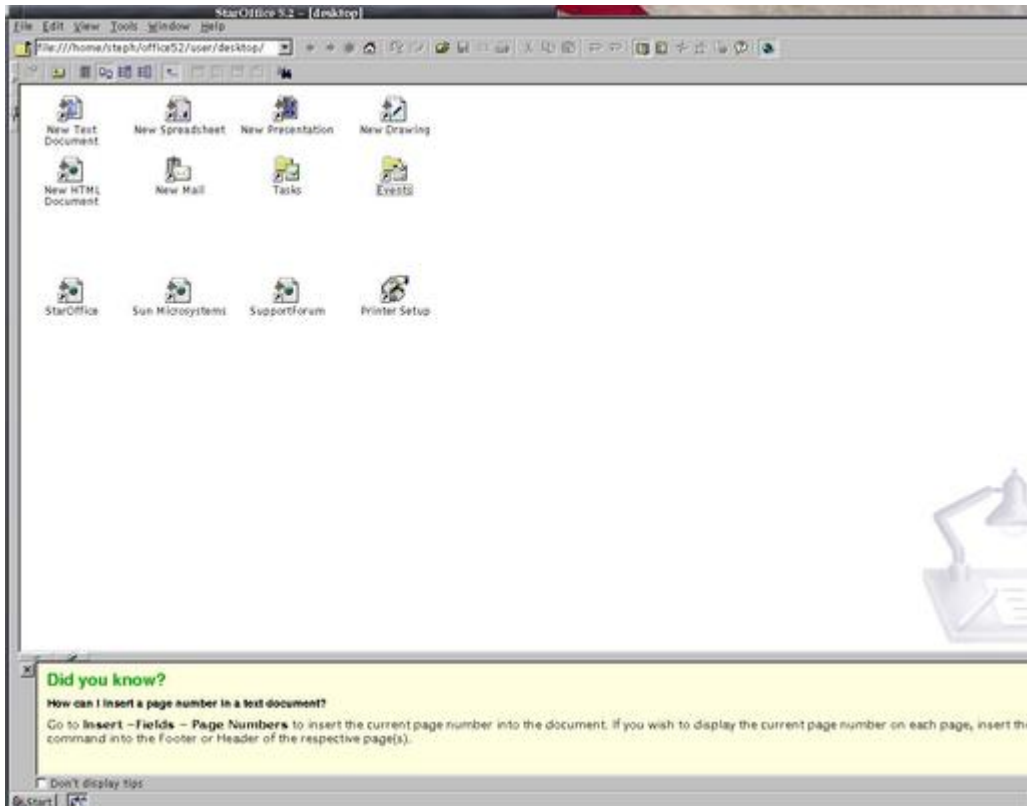


Figure 1. StarOffice Desktop

It is cause for concern that, by default, the “Integrated Desktop” option is enabled in StarOffice. A splash screen is something common enough with applications run in a GUI environment. A suite of applications taking over the desktop without the user's knowledge (or choice) is something else entirely. However, deselection of View --> Integrated Desktop (or Ctrl-Shift-I) will allow you to keep your chosen desktop and reduce the desktop to an application-size window.

To get started, you will need to look for a vertical grey arrow on the left-hand border of the desktop window. By opening Click & Go (see Figure 2) you will see a list of choices of what you can do.

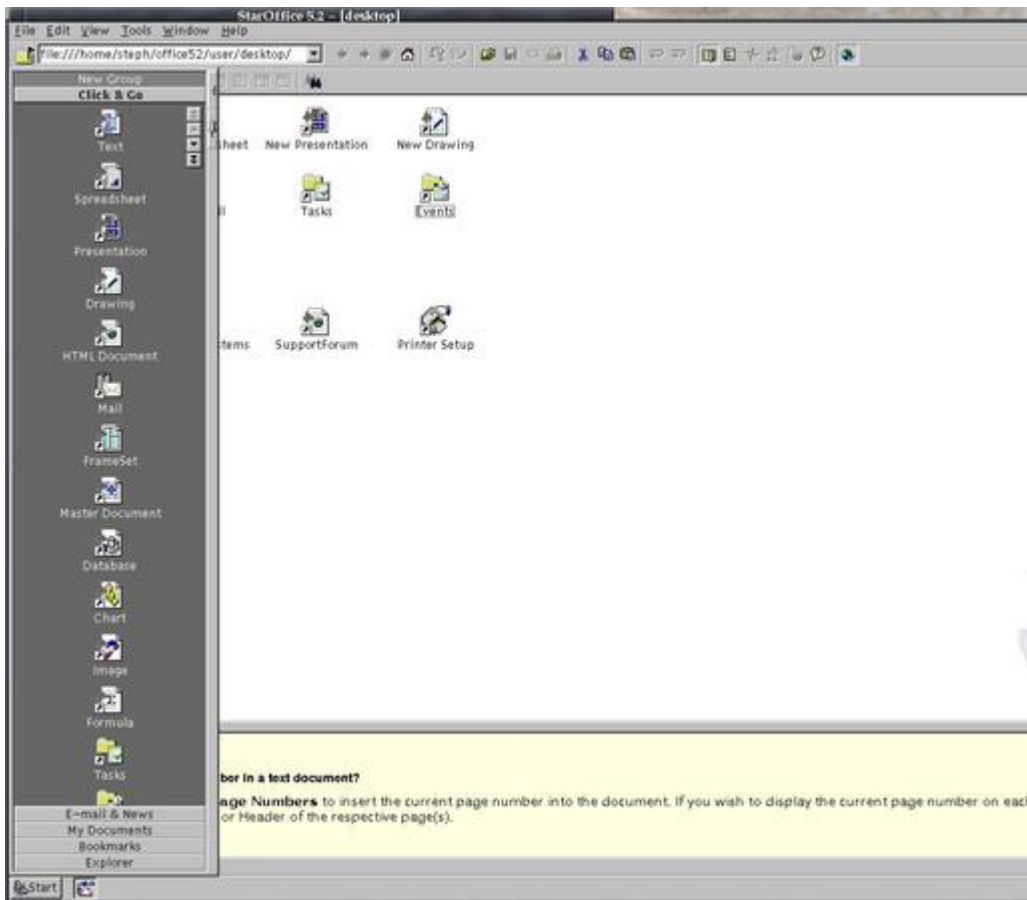


Figure 2. Click & Go

StarOffice Writer

As mentioned, the word processor functions all seem to be present and accounted for. Mercifully, there are some elements that exist and can (once you've closed the "correction" windows) be ignored, such as the tendency for the automatic spell checker to write "quantitative", when you were simply typing "quantity". The manual mentions "streamlining routine formatting tasks", which by all accounts involves turning on wordwrap as a default setting and not indicating this to the user.

My personal bias against HTML editors was confirmed upon attempting to draft a simple grey-on-black information page. SO 5.2 did not display the grey-on-black information page; it did display a black-on-black information page, which may indicate a certain independence of spirit to its creators but is hardly a useful quality in web publication.

There are several formats supported by Writer, including Word 6.0, Rich Text Format and StarOffice's "housebrand" of text formatting. ASCII is, thankfully, also supported—although you have to confirm this choice when saving your document.

StarOffice Mail and StarOffice Discussion

There are two ways to configure Mail and Discussion: you can specify that the configuration parameters should be the same as used in Netscape Messenger, and choose for the program to do this automatically, or you can manually enter these parameters. As I'm not altogether certain of the security risks involved in the first choice, I've chosen to do this manually. You can bypass these options by specifying "Don't Use Internet" with the click of a mouse.

E-mail is a nice thing to be able to access whilst pounding away on a recalcitrant document. Access you may, but replying to that proton epistle is a wee bit more of a challenge than one might wish, especially if "Reply" buttons/keystrokes are your custom. If all else fails (and it will), right click. The same principles apply to newsgroup-reading.

StarOffice Impress and StarOffice Player

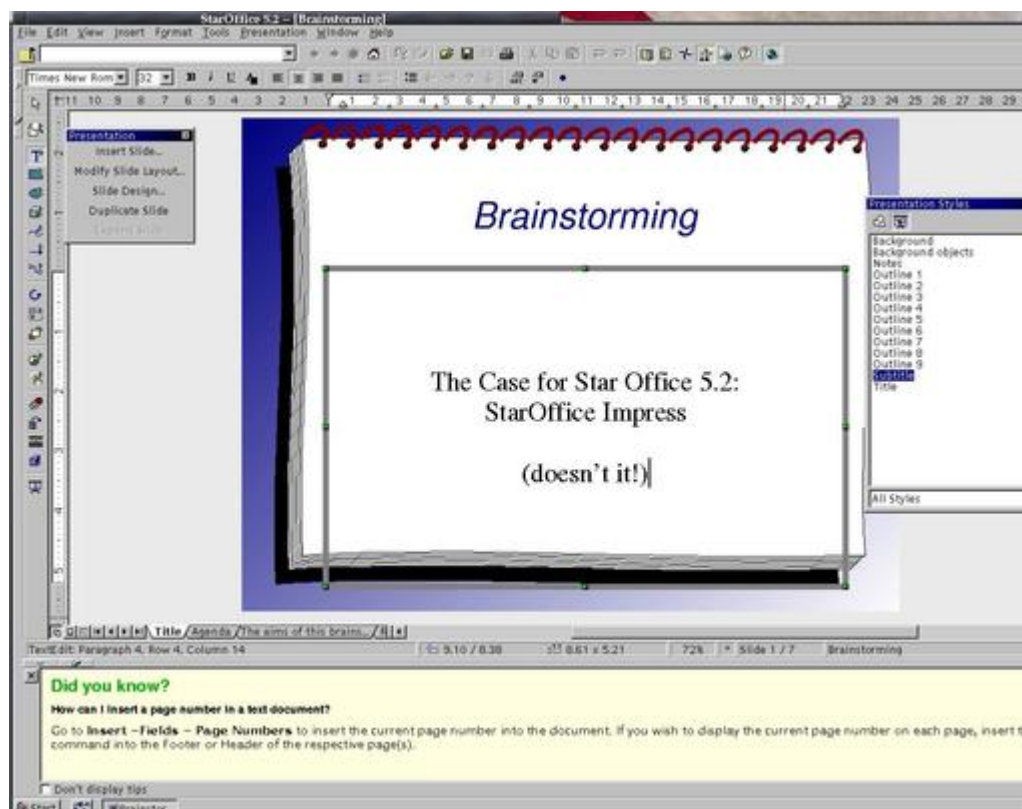


Figure 3. StarOffice Impress and StarOffice Player

Need to make a presentation to the board? Need to be able to rehearse timings of slides and make the best impression you can?

Both of these tools were surprisingly good. I'm no graphic artist but still managed to get a little slide up in less than seven minutes. Actual presentation makers could probably get the whole presentation done in that time. As it was, I attempted to make a few more "slides", create a slide show, change the order,

etc., and view the results with Player. In terms of tools that are designed for productivity, and not for messing about with configuring/de-configuring options, these two applications exemplify the ease of use the developers of other StarOffice applications should be aiming for.

Miscellaneous Information

There are minor components of SO 5.2 that are interesting and provide pleasant “dressing” to the office suite. Some of the clip art is quite well designed. The Schedule is very nicely set up, allowing the user to survey an entire week at a glance. Fonts, as well as 2-D and 3-D effects, make for some eye-catching products.

On 10/13/2000 (yes, a Friday) Sun made the code for StarOffice 5.2 available for download (<http://openoffice.org/>). It's worth noting that Sun is still retaining the copyright of the project, regardless of who contributes what code or how much of the code is completely rewritten.

Conclusion

From what this writer has seen, three elements are missing from Star Office 5.2: consistency, modularity and unity. There is no consistent quality among the applications: some work very well indeed, while others are “not-yet-ready-for-prime-time”. Modularity would allow each application to run separately, as does Player, and therefore give the user more direct control over what s/he wants to run. Running everything from “one place” sounds nice, but better results would arise out of an expanded context menu than from a bulky pseudo-desktop, which is more visible and serves less purpose than the menus concealed in its left border.

An office suite is normally categorized as “Productivity Software”. StarOffice 5.2 doesn't yet belong here.

The Good/The Bad

Stephanie Black is a recent migrant to IT and owns and runs Coastal Den Computing, a Linux consultancy. She has spent 80% of her coding life working with Linux and can be reached at alphafemale@radiant.net.

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

[Advanced search](#)

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

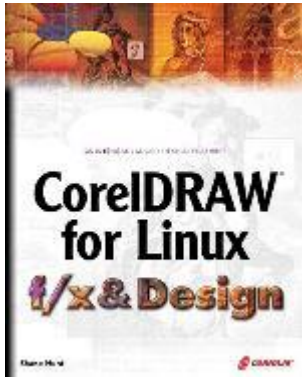
Advanced search

CorelDRAW for Linux: f/x & Design

Clifford Anderson

Issue #81, January 2001

No getting bogged down with print issues and page setup, this book is about creating.



Book Review

- CorelDRAW for Linux: f/x & Design
- Author: Shane Hunt
- Publisher: Coriolis Group
- URL: <http://www.coriolis.com/>
- Price: \$49.99 US
- ISBN: 1-57610-686-1
- Reviewer: Clifford Anderson

It is without doubt that most computer books take more time to read than the program takes to learn. What is worse, these books can tire even the most studious person to the point of deflating that pumped up feeling that accompanies tackling a new program.

Fortunately, such is not the case with *CorelDRAW for Linux: f/x & Design*, by Shane Hunt (Coriolis Group, 2000). Hunt is not the kind of author that merely tells you about the program, like so many glorified help-file-style books. He

writes to get the reader into the program. Well-written and slightly eccentric, the book will have the reader screaming for more with each chapter.

The title of the book is not misleading. Throughout the book, the author focuses on designing and creating great effects with CorelDRAW for Linux. No getting bogged down with print issues and page setup, this book is about creating. Coriolis Group calls their F/X Series of books the Creative Professionals Press. Hunt's delivery and content does not undermine the brand, but exemplifies it.

The book is made up of 18 chapters and has four to five projects per chapter, each of which stands alone in its own right. But, for the sake of his learning audience, he creates a linear movement, beginning with reasonably simple exercises before moving on to more complex projects. Let's take a look at one example from Chapter 2.

Shane gives a respectful but playful nod to his Linux readers by having us tackle creating our favorite mascot, Tux the Penguin. In 17 steps (less than a paragraph for each step), the student will have a hand made mascot right on their desktop. The fulfillment of this project comes from the fact that we made him from scratch. That's right. The head, body, flippers, feet, beak and eyes are all created by us, the readers (see Figure 1).



Figure 1. Tux the Penguin

How does he do this for us? The answer will reveal the content of Shane's book as well as what the reader can expect throughout the duration of his work.

Hunt teaches us like someone teaching a student driver. He's not going to describe every minor detail of the transmission before he allows us to get in the driver's seat. Rather, he says, "Here's where you sit, there's the steering wheel, the pedals, stick shift and windshield. Let's go." From here, he shows and tells you how to drive while you are driving. This understanding of how to write for an audience who needs to be doing work (rather than reading about it) is what elevates this book as one that does not belong on your bookshelf but on your desk.

In Chapter 2, Hunt himself sums it up for us: “You don't have to be dripping with talent to be a good artist; you just have to learn how to use the tools that you have and be patient enough to work out any design problems that arise.”

When learning a vector-based drawing program, everything begins with a simple shape: a line, a circle, a square. Likewise for Shane's method. He embarks on a trek that begins with these simple shapes (just like our penguin friend) and then, through acquiring the necessary familiarity, technical expertise and willingness to participate, we acquire a knowledge and skill base that allows us to go far beyond the confines of his book.

Exactly how much can one acquire from one of Shane's projects? Consider our penguin, above. Once done, the reader will have had hands-on experience with over a dozen of CorelDRAW's tools with an equal amount of techniques for manipulating its features.

Immersion by example is what Shane's work is all about. The projects he gives us have to be seen rather than heard. But a short list will provide an inkling of what to expect. From the perspective of how CorelDRAW for Linux works, the reader will be exposed to editing shapes, working with transparency and shadow, creating blends and contour effects, as well as incorporating Bitmap (raster) images and modifying them within CorelDRAW's environment. From a creative stance, Shane has us designing ads and banners, optical illusions, chains, blades, barbed wire, comic book characters and celestial wonders. All this by no means exhausts the book's content.

The overall structure of the book caters nicely to a learning environment: lots of screenshots to determine where we are in the program, lots of images of the work in progress to evaluate our own work, plenty of secondary detail alongside the figures to add clarity to whatever tool or method we are currently using, and enough white space to add our own comments or notes.

Shane abides by the principle “tell them what you're going to tell them, tell them, then tell them what you told them.” This proves to be an excellent heuristic device when going through the book's self-contained chapters.

Although Shane takes writing about CorelDRAW for Linux seriously, many of his examples may give the impression that his book is not to be taken seriously. If you're a technical illustrator who needs to know CorelDRAW as an occupational hazard, you may find yourself a little overly amused at some of the projects in the book. You're in need of creating a span bridge, and you're faced with a chapter on tattoos and barbed wire—consider it a challenge to expand your horizons to illustrate with diversity!

Most, if not all, of the book is about illustrating the fun stuff—things that look cool. This is Shane's trademark style and it shows—and works. But what is most important and justifies buying the book is its ability to have the reader master the immense girth of CorelDRAW for Linux while also becoming immersed in the program itself.

As a fellow purchaser of books, I am always asking the same question over and over as I thumb through the myriad publications for computer programs: what will this book do for me? And too many times the reply comes: not much. Such is not the case with this book. *CorelDRAW for Linux f/x & Design* by Shane Hunt, delivers. With every chapter and every project, the reader will be learning, using and mastering the well-known power that has made CorelDRAW a standard in vector-based illustration and design.



Clifford Anderson is a freelance writer and graphic designer. Currently residing in Portland, Oregon, he can be reached at aclifford@uswest.net.

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

[Advanced search](#)

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

[Advanced search](#)

Letters

Various

Issue #81, January 2001

Readers sound off.

Shallow Book Reviews

Sorry to have to say this, but I can't recall the last time I read anything in *LJ* that interested me. Business, web programming and networking take up the bulk of your publication, and I'm sure these things are of interest to many, but not to me. Furthermore, many of the software/book reviews are amateurish and almost all of them lack depth. For example, a review of a math program that actually gave an example of how to do something *nontrivial*, as opposed to "this program knows how to multiply matrices, isn't that neat" would be welcome. Useful tips for the garden-variety UNIX user would be nice, but that's already covered quite well by *Linux Gazette*. I also wouldn't mind seeing a well-written article that provided some hard criticism of the shortcomings of various Linux distributions and pulled no punches, but of course you don't want to offend your advertisers.

—Tom Goldring

Backup Recovery Revisited

I am an incarcerated subscriber. Until my recent transfer to a pre-release center I worked with computers for ten years in various industries within the Texas prison system. I have worked with Linux off and on over the past five years. Our computers have tended to be older machines handed down after their primary users deemed them obsolete. Thus, they are more prone to hardware failures. I have learned the hard way that backups are essential. I have come to appreciate a backup and restoration process that is as simple as possible—Keep It Short and Simple (KISS).

I read with interest Charles Curley's article "Bare Metal Recovery" (November 2000). And, I have to credit him with devising a sophisticated method of backing

up and restoring a system. Good work! However, his method is too sophisticated, and I balk at the thought of having to repeat such a complex process for each machine that I need to back up. The problem is that the restoration disk contains the backup of the base system ("metadata", as Mr. Curley calls it) and, therefore, a restoration disk must be built for each and every system. (Not to mention the need to resave the base system to the restoration disk every time the configuration is changed.) The restoration disk should only contain a minimal system with the appropriate drivers and utilities and the restoration software. The backup of the base system itself should be contained on the normal backup media. This way a single restoration disk can be used on any number of machines.

If you are using a backup utility that is too large to fit on a floppy, then building a restoration disk on a Zip drive (or Jaz drive or spare IDE hard drive) is appropriate. However, I noticed that the venerable tar utility won the Favorite Backup Utility category in your "2000 Readers' Choice Awards" (November 2000). Hurrah! tar is my favorite, too. Why? Because with very little effort I can restore a Linux system (or an entire department of Linux boxes) from a single set of boot and root floppy disks. You can build your own floppy-based system (re: "Customize Linux from the Bottom—Building Your Own Linux Base System", November 2000) or you can use any of the many miniature distributions. I prefer to use Slackware's boot and root disks (I may be a little out of date here, not having used their newer disks since version 4.0.). Their "scsinet" boot disk contains common SCSI and NIC drivers. If you need a driver that is absent, you can build a custom kernel from the source (same version, of course) and write it to the boot disk. Their "rescue" root disk contains tar, file system and network utilities. You may need to create a device node for your particular backup device. I appreciate the convenience of Slackware's boot and root disks. I have never had a need for anything else, and I have had many occasions to restore a downed system or migrate a system to another machine. tar is definitely the backup/restoration tool du jour and a floppy-based system is the easiest means of restoring one or many systems. KISS!

Sincerely,

—Antonio Barta

I read your October 2000 "Comparison of Backup Products" by Charles Curley with great interest. The article was very surprising, especially since our product (Microlite BackupEDGE) was not included in the test.

I say surprising because we've been in the Intel UNIX/Linux backup marketplace since 1987. Also, we were the first commercial vendor to introduce a disaster

recovery component for Linux products, back in April of 1999 at Comdex Spring.

In short, how did you miss us?

—D. Thomas Podnar, President Microlite Corporation tom@microlite.com

The only reason I did not mention your product was simply because I needed to limit the review in size and scope to keep it down to a manageable size. As you know, there are a lot of good products on the market, and plenty of gnuware and freeware products as well. Had I taken the time to cover all of them, I'd still be at it, and would take up an entire issue of the magazine if I ever got it done. I do hope that my comments on the products I reviewed were sufficiently general that the reader can learn something about evaluating backup software in general.

—Charles Curley

Victor Yodaiken on Real-Time Linux

Re your “Real Hard Time” article in the November issue of *Linux Journal*: your readers should know that both TimeSys and Lineo use real-time technology taken from FSMLabs RTLinux. What those companies add (other than a fascinating interpretation of the GPL) is a misunderstanding of what makes RTLinux useful. RTLinux adds a special multithreaded, real-time process to Linux and rigorously prevents Linux from delaying the operation of this process. The real-time process schedule, itself, ignores Linux synchronization, and has access to the raw hardware. Instead of adding “features” to the real-time process, we encourage developers to break out of the box and take advantage of a dual environment: hard real-time with a stripped down POSIX threads API in this special process, and the stability and power of Linux outside of the special process. The pedal-to-the-metal performance and reliability of RTLinux are possible because our design allows us to toss away many “features” that have been considered essential in previous real-time operating systems.

In my ever so humble opinion, some of the new arrivals to Linux and RTLinux operate under the mistaken theory that simplicity is a “lack of maturity” in the technology. But simplicity in both Linux and in RTLinux is a deliberate design choice and is the hard-won and hard-to-keep “feature” that we most prize. In horror movies, the victims have to walk down the steps into the dark basement to see exactly what is making that growling noise. Don't they know that IRIX and RT-Mach and who knows how many other victims have preceded them down the stairs—and none of them ever came back?

—Victor Yodaikenyodaiken@fsmlabs.com

Look for Victor and Edgar Hilton's article on RTLinux in the January issue of Embedded Linux Journal —Editor

Advertisement Uproar

Readers response to Qsol.com's advertisement in the November issue was vast and varied. Below are a few samplings from each side along with the advertiser's response.

—Editor

I think the picture of a naked programmer on the supplement was a bit much, but still laughable. However, in the November 2000 *LJ*, page 43 contains an ad by Qsol.com that contains what some would call vulgar insinuation.

Personally, I think this type of sexism has now gone too far in your magazine and sullies the character, professionalism and image of *LJ*, not to mention some of the people who buy it. I ask you to please contain the college crowd a bit more, and give us a magazine we can proudly display on our living room coffee tables or at work without having to rip pages out.

I for one will never use Qsol.com's products as I don't agree with their advertising message.

—Gordon Cunninghamgcunnin2@bellsouth.net

Not that I want to be accused of being politically correct of anything, but I bought my copy of *Linux Journal*, November 2000, and found the advertisement opposite page 42 a *leeeetle* beyond the pale. I mean, [that caption]--with that picture! I had a flashback to advertisements from the 1970s.

God knows we have few enough women in this industry (and in Open Source specifically) as it is, and adverts like this aren't going to help us any.

Not that I want to stop a corporate citizen's rights to "individual speech" or anything, but maybe someone might hit them with a clue stick?

—Jeremy Allisonjeremy@valinux.com

As the proud owner of 150 copies of the Python supplement, I read with rapt attention the many letters regarding its famous nudie cover. This has all left me very curious as to what kind of response you will get for the QSol ad on page 43 of the November 2000 issue. I hope you will print a sampling. It's very

interesting from a marketing perspective. I'm fairly sure you wouldn't see this stuff in *Byte* or *InfoWeek*, and the fact that you do see it in *LJ* tells me something about the markets Linux is and isn't in.

Not that I ever want to see you turn into *Byte*. It's just that I've worked in Corporate America and I can see where the negative response comes from. The part of me that tries to sell Linux/Python solutions to this often ultra conservative group feels that a print leader of the Linux community such as *LJ* needs to walk a more conservative line.

But, thankfully, the rest of me says to heck with it. Let someone else do *Byte* for Linux. I hope that whatever you do, you continue to print articles with so much delicious geeky detail and technically focused variety.

—Stephansdeibel@archaeopteryx.com

I would like to commend you on having the guts to print what is probably one of the most creative ads I have seen in a long while. The ad on page 43 of the November 2000 issue. A lesser magazine (especially one that just had problems with a cover shot that included a naked man—another awesome show of backbone on *LJ*'s part) probably would have refused it. I know you will probably take a lot of heat for it but, I am impressed. I thought the ad was a riot. My only problem is whether to tear it out and pin it up in my cube or keep the article together in the magazine. All things being equal I would much rather deal with a company with a sense of humor than one that refused a funny ad for fear of offending someone. In fact, all things not being equal I would rather deal with a company with this type of ad. Please keep up the GREAT work and don't back down from the people that will always complain.

—James Alspachjalspach@csione.com

We sincerely apologize to all those who have expressed concern about our advertisement recently featured in Linux Journal (November 2000). It was certainly not our intention to be offensive and we wish to again express our regret to anyone who was displeased by the ad. We understand that this has angered some readers and have therefore reacted immediately by pulling this artwork from all future issues of the magazine. Again, we extend our sincerest apologies.

—QSol.com

Source Known

The *LJ* for November 2000 had a “source unknown” quote in the column titled “Let Freedom Ping” by Doc Searls. A quote that bears remarkable resemblance to the one mentioned is thought to be a Kenyan prayer, listed in *Simpson's Contemporary Quotations*. Just my two cents.

Cheers,

—Tux@L7.net

Acronyms Explained

I love your magazine, except when you use strange acronyms without telling us what's going on. What is “URI” as used in Brian Aker's article (*LJ* page 172 November 2000). I've looked everywhere, including three years of back issues of *LJ*, and can't find this acronym used anywhere. Spare us the frustration and explain your acronyms every time, please.

(Don't tell me it's really “URL”!)

—Reilly Burkereilly@aerotraining.com

A URI is a “Uniform Resource Identifier”, and URLs are a subset of URIs. For more information on the difference between URIs and URLs, see <http://www.w3.org/Addressing/>. By the way, that was a URI, URI and URN all at the same time. We'll expand that acronym next time we use it.

—Don Marti, Technical Editor *Linux Journal*

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

[Advanced search](#)

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

[Advanced search](#)

UpFRONT

Various

Issue #81, January 2001

Stop the Presses, *LJ* Index and more.

***LJ* INDEX—JANUARY 2001**

1. Increase in XML usage among Linux developers over the six months prior to September 2000: 75%
2. Surveyed Linux developers using XML: 28%
3. Surveyed Linux developers who use XML up to half the time: 27%
4. Surveyed Linux developers who use Java: 54%
5. Decline in club meetings over the last 25 years: 58%
6. Decline in family dinners over the last 25 years: 33%
7. Decline in “having friends over” over the last 25 years: 45%
8. Reduction in your chance of dying if you join one club next year: 50%
9. Year by which open-source standards are expected to completely change the software industry: 2004
10. Number of unrelated Zelerate (formerly Open Sales) founders whose surname is Ferber: 2
11. Return on U.S. venture funds in the fourth quarter of 1999: 59.4%
12. Return on U.S. venture funds in the First quarter of 2000: 23.1%
13. Return on U.S. venture funds in the Second quarter of 2000: 3.9%
14. New funds raised by venture capitalists through mid-October of 2000: \$64,000,000,000 US
15. Projected percent return on venture funds over 20 years, as of 6/30/2000: 19.9%
16. Share price once predicted for Priceline by Merrill Lynch Internet Analyst Henry Blodget: \$150 US
17. Priceline share price on April 22, 2000: \$76.38 US
18. Priceline share price on April 30, 2000: \$165 US

19. Priceline share price projected in May of 2000 by Jamie Kiggen, then of DLJ: \$190 US
20. Priceline share price on October 17, 2000: \$5 US

Sources:

- 1-4: Evans Data Corporation (September 26, 2000)
- 5-8: BowlingAlone.com
- 9: *Wired*, quoting Forester Research
- 10: Zelerate
- 11-13: Venture Economics
- 14: *San Jose Mercury News*, quoting Venture Economics
- 15-20: CBS.MarketWatch.com

THEY SAID IT

The work of thought is one of the most ancient and useful activities of humankind. To generate thought is to create life, liveliness, community. Consensus isn't important. What's important is how the generative power of our thought makes life vivid and burns out the dead brush, dead thoughts, dead institutions.

—Michael Ventura

Genius is applying the originality of youth to the experience of maturity.

—Michael Polanyi

Documentation is like sex: when it is good, it is very, very good; and when it is bad, it is better than nothing.

—Dick Brandon

Idealism is fine, but as it approaches reality the costs become prohibitive.

—William Buckley Jr.

Crappy old OSes have value in the basically negative sense that changing to new ones makes us wish we'd never been born.

—Neal Stephenson

Believe it or not, the Internet is actually under-hyped...We are coconspirators in the largest, legal creation of new wealth, primarily in Internet companies.

—L. John Doerr

Perpetual growth is the creed of the cancer cell.

—Edward Abbey

Simple, clear purpose and principles give rise to complex and intelligent behavior. Complex rules and regulations give rise to simple and stupid behavior.

—Dee Hock

The cracked ones let in the light.

—Tom Peters

The only way to be truly creative is to never repeat yourself.

—Rob Breszny

We don't have a good language to talk about this kind of thing. In most people's vocabularies, design means veneer. But to me, nothing could be further from the meaning of design. Design is the fundamental soul of a man-made creation.

—Steve Jobs

The corporation as we know it, which is now 120 years old, is unlikely to survive the next 25 years. Legally and financially yes, but not structurally and economically.

—Peter Drucker

The greatest danger for most of us is not that our aim is too high and we miss it, but that it is too low and we reach it.

—Michelangelo

STOP THE PRESSES: OPEN SALES BECOMES ZELLERATE

Once again we're writing about something before it's happened, for publication afterward. Dig this: where I am, it's still October, the windows are wide open on the 26th floor above West 56th in Manhattan, and the leaves on the trees in Central Park are just starting to turn color. Where you are, every one of those leaves is mulch and *Open Sales* has long since morphed into Zelerate.

Which is what really matters, for this story. The leaves fall every year. *Open Sales* can shed its name but once.

It makes sense, of course. *Open Sales* was founded by two unrelated guys named Ferber (Coincidence? Yes). And a quick check of Google (whose collection of umpty-thousand Linux machines has intimate knowledge of 1,247,340,000 web pages) discloses some persuasive statistics:

- open = 31,000,000
- sales = 18,100,000
- open sales = 801,000
- "open sales" = 5,180
- zelerate = 0

In other words, *Open Sales* was a good example of applied generics. You couldn't ask for better camouflage if your name ended with "Solutions".

So what's with Zelerate?

Paul Carlstrom, the company's director of corporate communications, says this: "Rob and Glenn (Ferber)...cofounded *Open Sales* to offer open-source e-commerce applications in 1998. A lot has occurred since that time: The GPL'd software has caught the attention of a growing cadre of developers. The product has matured rapidly. The installed base of *Open Sales AllCommerce* has risen..."

Soon, the company found itself doing "accelerated commerce solutions". And, "because we feel you can't get from (a) to (z) without knowing where (z) is located, we've chosen the last letter of the alphabet to start our name."

Carlstrom also wants us to know that open source is what primarily accounts for development speed—for two reasons.

One is that "we have expanded the open-source model to include customer-mandated direction as part of our development process."

The other is that Zelerate isn't a product company. it sells service. With enterprise-class applications, he explains, "e-commerce packages can cost in the hundreds of thousands of dollars, and the customer will still end up paying more for services and support than for the original licensing fees." By not making development secretive, Zelerate allows constant customer input to the development process.

There is a common belief that service companies can't grow (or, to use last year's word, *scale*) as fast—or as large—as product companies. This may not be a bad thing in any case, but Zelerate is now named to test the theory.

—Doc Searls

Erratum--

The references to IMAP in Mark Stacey's article, "Linux as a Work Environment Desktop", (issue 79, November 2000) should read MAPI rather than IMAP.

LJ History

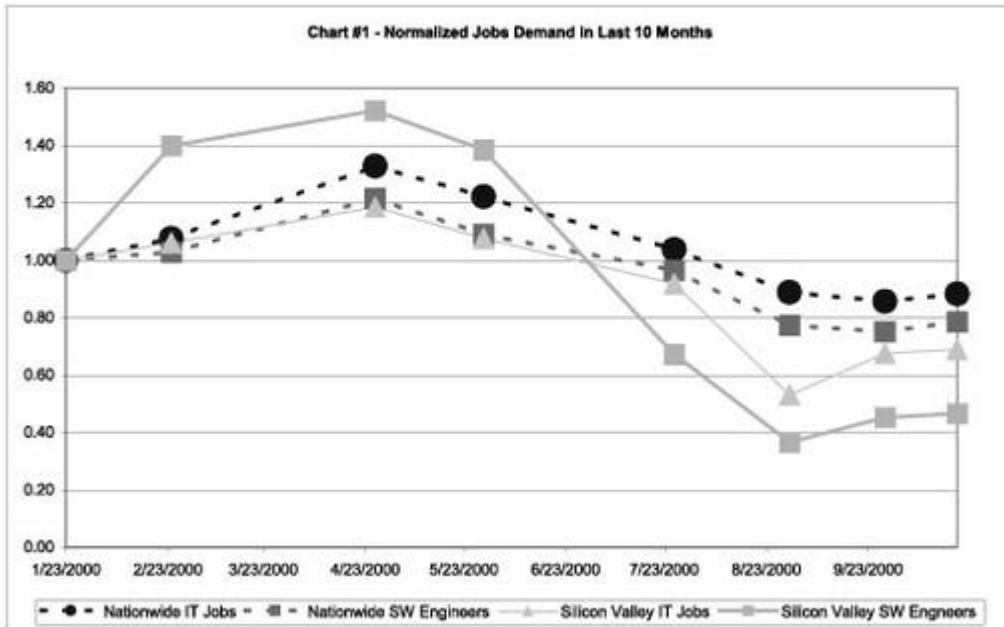
In *LJ* six years ago this month we published "A Conversation with Linus Torvalds" in which Linus reflected on being bitten by a penguin in Canberra, his preference for Irish beer, getting ready for 1.2 and his decision to name the OS Linux rather than Freakix. Linus wrapped up the interview by revealing his plan for "World domination. Fast."

Trends

by Reginald Charney

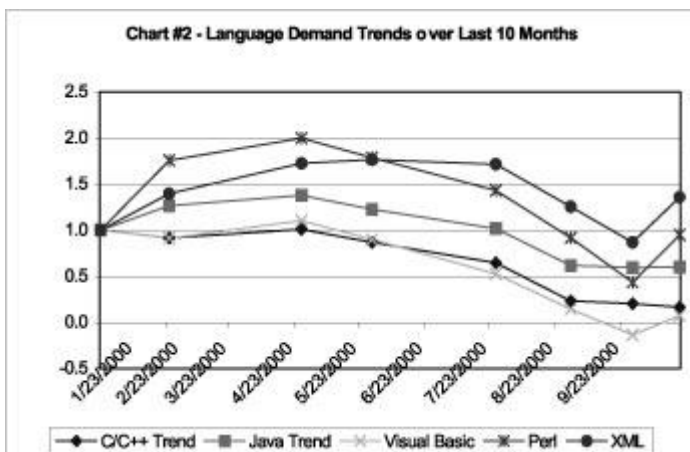
Job Openings

Last month, there was a slight up-tick in the number of software engineering jobs in demand, particularly here in the Valley. This month proves that it was no fluke. There is still a continuing demand for both IT skills in general and software engineering in particular. However, it does look like it is leveling off. The demand now for software engineers appears to be increasing at the same rate as for all IT positions. (Note: Chart #1 is normalized for the number of jobs in January of this year. That is, the number of openings in January 2000 has been taken as 1.00.)



Language Demand Trends

The demand for all the major computer languages have shown an interesting set of trends since the start of this year. Chart #2 shows the demand for today's most popular computer languages. The demand peaked in the April-May time frame when demand overall for language skills was at its highest. Since then, demand for language skills has decline until last month when things turned around. Demand for the Internet-related languages, XML and Perl have increased while the demand for Java and C/C++ has leveled off. My conclusion is that the desire to connect systems together using shared data definitions and universal scripting languages like Perl is driving this demand. Visual Basic is the anomaly. Demand for it increased, probably because of new prototyping being done.



Reginald currently heads the US chapter of the Association of C and C++ Users.
Visit their site at <http://www.accu-usa.org/> to learn more.

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

[Advanced search](#)

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

[Advanced search](#)

Editors' Choice Awards

LJ Staff

Issue #81, January 2001

And the winners are....



As reported in the 1999 Editors' Choice Awards, it's been another year of bursting developments and blossoming advancements for our favorite operating system. Not only is Linux becoming the hot OS of embedded environments, but the first year of the new millennium has witnessed the expiration of the RSA patent, allowing open-source security tools to be included in Linux distributions. The SourceForge site, furnished by VA Linux has quickly become perhaps the most valuable resource for programming tools and information. More attention than ever is being paid to the penguin OS as a desktop alternative by large corporations and venture capitalists who are finally beginning to understand the free business model and are, happily, coming to accept the GNU GPL instead of coming up with their own incompatible licenses. Additionally, an increasing amount of previously proprietary software is becoming open source, such as Sun's StarOffice, now free under the GNU GPL. All this translates to an ever-enlarging field of competitors for the Editors' Choice Awards. And not only are there more entries than ever, but almost all entries are increasing in quality and functionality, making the choice of our favorites akin to deciding on a drink at an open bar with a one-drink limit. Sure you can have that Tom Collins, but only at the expense of the White Russian. If some of our choices rub you wrong, make it a stiff one and be reconciled.

+Server Appliance: Cobalt Qube 3



We know, we know, we should have picked the rackmount version of this easy-to-configure server. The "RaQ" is the one that people have been buying by the truckload. But we like the cube shape. It's so...cubical. The whole thing just works, even for people who aren't Linux experts. It was the happy, new users who clinched our decision in favor of this box. Read the QubeQuorner weblog if you want to see some of the stuff that the user community (Quommunity?) is up to.

+Web Server: VA Linux Systems 2200 series with Debian

VA Linux Systems has a good reputation for getting good hardware and installing Linux on it. Last year, they got ambitious, designed their own easy-open 2U chassis, and filled it with hot-swap RAID drives and other quality parts. This year, the puzzle is complete, as they put Debian on the latest 2200 series servers. Now you can upgrade your software as easily as you swap a drive.

+Enterprise Application Server: IBM eServer zSeries

Everybody has been jumping up and down at how clueful IBM has become in the Linux department. But why change the name of the System/390? That's like renaming the battleship *Iowa*. But silly name or not, you have to love a server that lets you start up a whole new instance of Linux for a project without even going down to the server room to run another Ethernet cable.

+Office Workstation: ASL Merlin 60

Except for a loud fan in the preproduction system we tested, we like this box a lot. It's put together well from top-quality components and supported by a small, friendly company. The CPU is AMD's Duron, the cheap version of the celebrated Athlon, and we like the well-supported Matrox video card and the reliable IBM hard drive too. Just the thing for running the new, free StarOffice, or designing your new web site.

+Technical Workstation: Elinux.com

Frankly, nobody has yet shown us a high-end Linux workstation that has really knocked our socks off. The Linux workstation industry is still in its infancy, and good 3-D support still requires some tweaking and RTFM-ing. So, socks still firmly on our feet, we visited elinux.com, browsed their selection of Linux-compatible parts and built our own workstation. Maybe next year there will be a workstation that impresses us right out of the box, but we'll still like elinux.com for important stuff like SCSI cards and name-brand RAM.

+Web Client: the Internet Junkbuster Proxy

Early this year, we heard quite a bit about privacy abuses by the web advertising agency Doubleclick. So rather than getting “pushed, filed, stamped, indexed, briefed, debriefed or numbered” by their global-tracking database, we've installed privacy protection software: the Internet Junkbuster Proxy.

If you travel with your laptop, put a copy on it too. Not downloading banners can make hotel dial-up speeds tolerable.

+Graphics Application:(tie) [gif2png](#), [pngcrush](#)

Neither of our two choices for best graphics application even has a GUI interface. Sometimes you need to script something, and converting any unlicensed GIFs you may have lying around to the legal PNG format is one of those times.

gif2png includes a script to grovel through your entire site and expunge the GIFs once and for all. And, some software creates oversized PNGs when you “save as PNG”, and **pngcrush** can go back and do it right.

+Backup Utility: [Amanda](#)

There is a large selection of backup tools for Linux, and we're glad, because it makes it easier to back up your new Linux box where the Powers that Be have already chosen the backup software to be used.

At *Linux Journal* though, we run Amanda. We can get whatever software we want at no charge because software companies give us copies to butter us up. But our sysadmin team chose Amanda, which stands for “Advanced Maryland Automatic Network Disk Archiver”. There's an unrelated project called Amanda at Berkeley, so be sure not to install the wrong software—the Berkeley Amanda is the Antarctic Muon and Neutrino Detector Array, and you don't want to find yourself with a bunch of muon-tracking data instead of backups of your files.

Amanda is classic, old-school free software—infininitely configurable, not fancy, but reliable. It takes a while to learn and set up, but once you've got it configured the way you want, it should be trouble-free.

+Communication Tool: Jabber

In an earlier article we wrote, “Jabber is built so wide open that it's hard to see the limits of what can be done with it.” And we like the “explosive combination” of XML and instant messaging. With Jabber, any two identities, whether human or machine, can send and receive real-time messages that contain pretty much anything and can do so in a structured way, independent, if necessary, of intermediating protocols. Jabber is going to be the basis of some fun applications.

+3-D Application: Blender

Think of Blender as the planter for the render farm—it's the application that lets you design 3-D scenes. But fire it up and you'll quickly realize that Blender is no Xpaint. This is some heavy software and takes a real-time investment to learn. But we judge a tree by its fruits, and digital art produced with Blender is pretty impressive.

+Database: PostgreSQL

Tim Perdue, a well-known PHP wizard, explained the benefits of using PostgreSQL as a back-end database for his web projects: “To start, we will make use of Postgres' SELECT...FOR UPDATE syntax, which effectively locks selected rows so you can update them and commit your changes within a transaction.”

And, he adds, “With some databases, like MySQL, you can't easily lock specific rows of data to prevent other processes from decrementing the inventory while you are also trying to decrement the inventory. What you wind up with is inaccurate numbers and a useless inventory count.” Now, we can't have that. If your e-commerce site sells the last Furby twice, you're in big trouble.

For “big-database” features and the configurability and administration advantages of open source, we give PostgreSQL a big thumbs-up.

+Game: *Soldier of Fortune*

If you invest your time and money in a computer game, you want to see people suffer when you shoot them, right? Anatomically correct gunshot wounds. Screams of agony. Simplistic plot. Put it all together and what do you get? Hours of fun, that's what.

Weak stomach? Try GNOME *Freecell*.

+Development Tool: Python

Hell is other people's Perl. Enough said.

Office Application/Suite: StarOffice

We must confess something right now. Here at *Linux Journal*, we're just not big office suite users. We basically get by with vi until we send articles to layout, and when someone sends us a proprietary-format document, we just trash it and add them to the bozo list. But every once in a while, in our personal lives, we have to do up a batch of anti-DMCA flyers or something. For that, there's StarOffice. A bit on the hoggish side, but it gets the job done. And we'll buy beers for Sun people at the next Linux show—thank you for putting the new version under the GPL.

+Desktop Environment: KDE

Remember how Matt Suhey didn't rap on the "Super Bowl Shuffle" but he gained a bunch of yardage, including a touchdown, in Super Bowl XX? KDE is the Matt Suhey of the Linux desktop. Despite the attention paid to a certain other desktop project this year, we're most impressed with the stability and just plain well-thought-outitude of KDE. KWord is a promising word processor, and all the desktop doodads work well too. Most of us at *Linux Journal* run KDE.

Troll Tech ended up fixing the KDE licensing mess with the stroke of a pen, but the next licensing controversy could go the other way, if the company involved turns out to be less cool than Troll Tech. So, kids don't try the license incompatibility thing at home.

By the way, that's Super Bowl 0x14 for those of you who aren't NFL fans.

+Mobile Devices: Tuxtops



Tuxtops is in a strange position, halfway between being a hardware vendor like VA Linux Systems and being a *Consumer Reports* for Linux laptops. They're too small to influence manufacturers' hardware selections, so they have to take the best laptop they can get and put customized software on it. But Tuxtops is a refreshing burst of honesty in a shrink-wrapped world.

We like their "Coming Clean" pages, in which they list the inadequacies in the hardware they ship. One page points out that one system's Lucent modem is "cranky and temperamental", even with the supplied Linux driver, and advises buyers to "ignore this hardware and consider it vestigial". Thank you; I'll get the optional PCMCIA modem instead of wasting my time. Funny, the more bad things you say about your products, the better you look.

+Embedded Development Tool: Microwindows

The Microwindows demo was one of our favorites at LinuxWorld. Imagine a GUI project that includes X- and WinCE-compatible APIs, alpha blending, proportional fonts, handwriting recognition, a VNC client, a Minesweeper clone and more. Now imagine it in 100K. Can you say Linux PDAs? Better put a waterproof cover on them; we're drooling.

+Real-Time Tool: Red Hat's RedBoot

RedBoot is an embedded debug and bootstrap tool for running embedded Linux systems on embedded platforms including ARM, MIPS, MN10300, PowerPC, Hitachi SHx, v850 and x86. It supports booting from flash or from the network.

RedBoot provides ways to address real-time timing requirements that allow an application to respond quickly to real world needs. It also provides some important tools to debug in this environment, which in our opinion is an extremely important issue. It is also completely open source.

+Toy: CueCat

Meow! This free bar code scanner was handed out at Radio Shack and included with some magazines (not ours). Not only did people dissect the kitty to disable its serial number (see the November 2000 issue) but they also wrote drivers and decoders to use it for all kinds of things, including cataloging their vast book collections.

The CueCat's manufacturer got into the fun by having their lawyer send out some of the most pointless and ludicrous threatening letters I've ever seen, which naturally made everyone get more CueCats to find out what all the fuss was about.

Apparently the original purpose of the CueCat was to get people to scan magazine ads instead of typing URLs, which must save some companies the trouble of learning HTTP Redirects. But, out of dumb business models come nifty toys.

+Book: *The Linux Network Administrator's Guide, Second Edition*

The *Linux Network Administrator's Guide*, by Olaf Kirch and Terry Dawson, has been a “living document” from the Linux Documentation Project since 1993 and still contains one of the best introductions to TCP/IP we've ever read. The new edition, released this year, is relevant to more Linux users than ever, since more and more of us are getting broadband Internet connections and setting up home networks instead of just using a PPP dialup. For users looking to take advantage of their DSL or cable connections, we recommend this book, which is available from the Linux Documentation Project web site.

+Web Site: Advogato

Once there was Usenet News and Linus Torvalds announced Linux there. Then many threads of discussion moved to lists, and web discussion boards, but forum after forum of intelligent conversation was stormed and destroyed by flamers and idiots. This year's chosen web site, Advogato, has been surprisingly jackass-free and has given us thoughtful discussions on password-protected textbooks, GNU autoconf, Bill Joy and choosing the right C data types to make your code 64-bit clean. Advogato is ruled by the cold equations of a reputation-ranking system, which has, so far, worked to protect it from the tide of drivel that washes over the rest of the “community” sites.

As a bonus, the Apache module that powers Advogato is fast, free and in our humble opinion, elegant.



TiVo, TV your way.™

+Product of the Year: TiVo

If you think Linux isn't ready for end users, well, put the TiVo in your pipe and smoke it. It's this year's hottest home theater component—a Linux-powered box that actually lets you “pause” and “fast-forward” live TV. No more waiting for a commercial to get another beer.

Resources

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

[Advanced search](#)

Advanced search

Best of Technical Support

Various

Issue #81, January 2001

Our experts answer your technical questions.

Can't Create User in COAS

I am facing a problem with COAS. I can't open the user creation tool. I deleted all lock files but still can't open this tool. What is the problem? —Anil Nair, anicmair@usa.net

This is a well-known issue covered in FAQs. Basically, you need to remove **/etc/shadow-** and **/etc/ptmp**. For details, do a search for COAS at <http://support.calderasystems.com/>. —Andy Bradford, andyb@calderasystems.com

Installing WordPerfect from CD-ROM

I have a Wordperfect CD and am trying to install it. I have made a Wordperfect subdirectory. Then I issue the following command, to begin installation:

```
darkstar:~/wordperfect# mount/dev/cdrom/cdrom
```

The error message I receive is as follows:

```
bash: mount/dev/cdrom/cdrom: No such file or directory
```

I switched to the CD-ROM subdirectory and issued the following command:

```
darkstar:/cdrom# mount/dev/cdrom/cdrom
```

--Larry Carnahan, lcarn876@aol.com

Are you sure **/dev/cdrom** is a symlink to a real CDROM block device such as **/dev/hdcx** ? Try:

```
ls -l /dev/cdrom
```

--Pierre Ficheux, pficheux@wanadoo.fr

Try this: **mount /dev/cdrom /cdrom** instead of **mount/dev/cdrom/cdrom**. Mount is the command, **/dev/cdrom** is the device and **/cdrom** is the directory. Between these three entities, there should be a space. —Paulo Wollny, paulo@wollny.com.br

Setting the ServerName for Apache

I installed an apache 1.3 from a Debian package. I used the command **apachectl start** and it sent me the following message:

```
Cannot determine local host name. Use the ServerName directive to set it manually.
```

What is the ServerName directive? I found a line about ServerName in the httpd.conf; is that what the message meant? I tried to set ServerName to **root**, then **127.0.0.1**, then **localhost**, then **daniel54** (the name of my machine at the prompt): nothing works. Apache gives no useful message; it just tells me httpd won't start. Is there a way to deactivate the Ethernet card without removing it physically? —Daniel Meilleur, daniel_meilleur@hotmail.com

You're on the right track—ServerName is indeed in httpd.conf. You need to give it a name to run as that matches a DNS entry. The easiest way to do that is to set up **/etc/host.conf** with an **order hosts,bind** line to force it to look in the hosts file first. Then set up **/etc/hosts** with a line that reads **127.0.0.1 localhost daniel54**. You can deactivate a network card at any time by typing **ifconfig eth0 down**, replacing eth0 with the appropriate value if you want to shut down eth1 or higher. —Chad R. Robinson, crobinson@rfgonline.com

Linux as Dedicated Citrix Client

Is there any method of installing Linux to operate as a Dumb Terminal so that the PC boots into Linux, auto logs on and launches a pre-configured application?

My idea is to configure some Thin-Client workstations using Linux as the OS and Citrix ICA client as the only application.

Ideally, the PC would boot to Linux, log in as a dummy user and launch the Citrix ICA client. —Stefan Ostadal, sostadal@gate.co.uk

You can do this simply by understanding how INIT works. When you boot your system normally, INIT will read **/etc/inittab**, which tells it to execute certain scripts. It also tells it to start a few getty processes, which allow you to log in. All you need to do is add an entry that runs your application, rather than getty.

Note that INIT itself is already logged in as root, so there's no need to execute log in in this case. Please note that this is a terribly insecure method of operating, so it's advisable ONLY for your specific case—a thin client for which local security means nothing. —Chad R. Robinson, crobinson@rfgonline.com

Dual Boot with Windows98

I have 4.0 gigabyte HD that is running Linux right now. How can I install Windows98 and have a dual boot system? —Navin Maahdkar, nrm@mindpsring.com

The easy way would be to boot with a Linux or Windows CD (or disk) and use fdisk to create a FAT32 partition for Windows and ext2 and swap partitions for Linux. You could then install Linux and Windows98 in the order you prefer. One warning though: if you plan to install Windows after Linux do not forget to create a Linux boot disk in order to reinstall LILO in the MBR and be able to dual boot again. The Windows installer will erase it. The “hard” way would be to use Partition Magic to rearrange the +space without having to reinstall Linux again. Be sure to make a backup before doing either. —Mario Neto mneto@argo.com.br

You need to have some space on your HD to install Win98 in the first partition. If not, you should install Win98 on another disk defined as master (or re-install everything!). Then you should configure LILO to set up dual boot. Something like this in the `/etc/lilo.conf`:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux
image=/boot/vmlinuz-2.2.14-5.0
    label=linux
    initrd=/boot/initrd-2.2.14-5.0.img
    read-only
    root=/dev/hdb1
other=/dev/hda1
    label=dos
```

Please check out the LILO mini-how-to to get more info. --Pierre Ficheux, pficheux@wanadoo.fr

Linux Server for a Home Network

I want to configure RHL 6.1 both as a server for a small home network (eth0) and as a dial-up client to my > ISP using dhcp (ppp0). What should my network configuration (i.e., DNS, gateways, routes, etc.) look like? —Tom Dolan, tdolan@erols.com

You need several things. We have little space here to explain everything, so instead I will point you to the documents that should clearly explain each step: 1.- Set up your LAN: www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Networking-Overview-HOWTO.html www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Net-HOWTO.html 2.- Hook up to your ISP: www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/ISP-Hookup-HOWTO.html 3.- Set SAMBA (if you have Windows PC's on your local net): www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/SMB-HOWTO.html www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Windows-LAN-Server-HOWTO.html 4.- Some security won't hurt: www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Security-HOWTO.html—Felipe E. Barousse Boué, fbarousse@piensa.com

More LILO Troubleshooting

I just installed Red Hat 6.2 and it worked fine. When booted, my system prompted LILO and boot Linux in Red Hat. I changed LILO to boot DOS. Now when I boot my system I get a C prompt. How do I get my Red Hat back? — Pablo, sonicmaster@teleweb.pt

You should use your Linux boot floppy to reboot under Linux. Then edit your lilo.conf to set up dual boot and validate by typing **lilo** as root. For example:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux
image=/boot/vmlinuz-2.2.14-5.0
        label=linux
        initrd=/boot/initrd-2.2.14-5.0.img
        read-only
        root=/dev/hdb1
other=/dev/hda1
        label=dos
```

--Pierre Ficheux, pficheux@wanadoo.fr

Windows' Clients Can't Find SMTP Server

I am having a problem with my end users trying to receive and send their mail. My mail server is working under Linux Op. All users receive the message, **The server could not be found (Account "Sendmail" SMTP Server; mail, Error number 0x800ccc0d)**. —Walter Minja, waminja@yahoo.com

Because of the error message you mention, it looks to me like your users are using a Windows client. You need to properly set up. For outgoing mails, set the SMTP IP address on each PC to the IP number of your Linux box in your local area network. For incoming mails, the POP IP address should be set up with the

same IP address of your Linux box. Bear in mind that POP3 service must be installed and enabled (with an uncommented "pop-3" line in `/etc/inetd.conf`). — Felipe E. Barousse Boué, fbarousse@piensa.com

rawrite Won't

I cannot get the rawrite program to work on any of my systems. I tried the autoloader from the Corel distribution, but all I am able to do is get to the point where they give the option to either install from the CD or create a boot disk. When the option to create a boot disk is selected, there is an error of some sort with the system not being able to transfer the boot image to the floppy. When the option to install from the CD is chosen (I have ensured that the computer's bios has the CD boot option selected), it reverts to the other boot options (floppy or C: drive). I have tried executing the rawrite program after booting my system with a Win95 boot disk but have no luck with either distribution.

My computer systems are all intel cpu-based, and the distributions I have are the i386 distributions. —Robert Cordner, clevon@hotmail.com

Assuming all files were downloaded (transferred) correctly, to generate a boot disk, the only file you need is the one named **boot.img** or **bootnet.img** (the latter for installs through a network). Transfer this file with ftp using BINary mode. After you get this file, which is pretty small (around 1.4 Mbytes), from DOS use the command: **rawrite**, which will ask you for the file name to write to the disk, prompt you to insert a disk and write the file you just downloaded to the disk generating an install disk. Of course, the rest of the distribution's files would be needed, so, download them all using BINary transfer as well, or if you use the bootnet.img image file, you may be able to install through a LAN if you are on one. —Felipe E. Barousse Boué, fbarousse@piensa.com

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

[Advanced search](#)

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

[Advanced search](#)

New Products

Heather Mead

Issue #81, January 2001

SuperServer 6010 and 5010 Series, GN 1600 Switchless Switch, Infinistore Version 1.2 and more.

SuperServer 6010 and 5010 Series

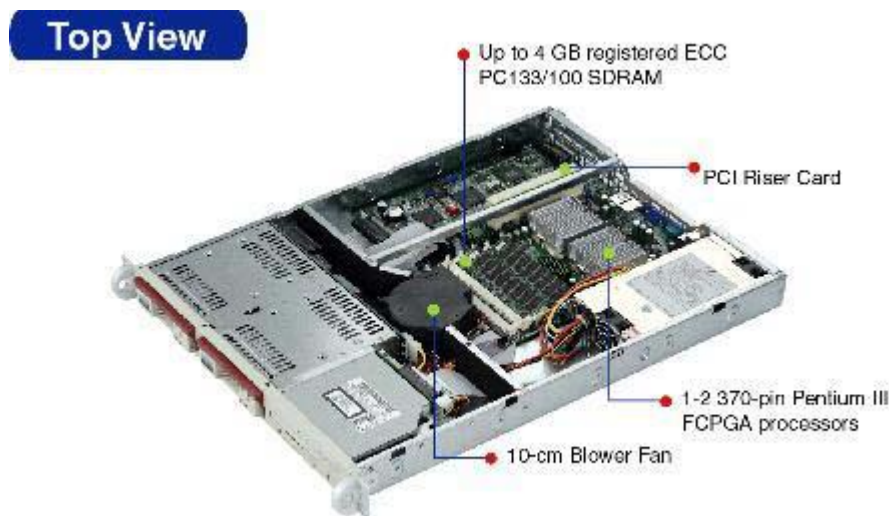


Figure 1. SuperServer Top View

The SuperServer 6010 dual processor and 5010 single processor series of 1U servers are available from Supermicro. Both series utilize dual onboard LAN controllers, provide fault tolerance, fast Ethernet and adaptive load-balancing solutions for broad-bandwidth network environments. The SuperServer 6010H utilizes dual Intel 370 FCPGA Pentium III processors up to 1GHz, ServerWorks ServerSet III HE-SL chip set supporting a maximum of 4GB of two-way interleave PC133/100 SDRAM, 64/32-bit 66/33 MHz PCI bus, Adaptec's Ultra 160 dual channel SCSI and Intel's dual 82559 NIC network interface capability.

Contact: Supermicro, 2051 Junction Avenue, San Jose, California 95131, 408-895-2000, marketing@supermicro.com, <http://www.supermicro.com/>.

GN 1600 Switchless Switch



Gotham Networks introduced the GN 1600 Switchless Switch that eliminates the need for a centralized switching fabric. Featuring scalability, centralized service provisioning and pay-as-you-grow pricing, this switching architecture decreases network design constraints and aids with scaling a network for new services and speeds. The universal service card (USC) captures switching, control and protocol processing on a single, multifunction board. USCs support IP and MPLS routing, ATM and frame relay switching and circuit emulation. The GN 1600 can support up to 16 USCs.

Contact: Gotham Networks, 15 Discovery Way, Acton, Massachusetts 01720, 978-263-6890, info@gothamnetworks.com, <http://www.gothamnetworks.com/>.

Infinistore Version 1.2

Grau Data Storage, Inc. announced version 1.2 of its Infinistore Virtual Disk (IVD) network attached storage server that provides disaster recovery capabilities and remote administration services. It combines hard disk RAID storage capacity, ultra-capacity tape storage and integrated storage management software. Multiple IVDs can be managed at a single location or distributed globally.

Contact: Grau Data Storage Inc. (USA), PO Box 271030, Louisville, Colorado 80027, 303-665-3018, info@GrauData, <http://www.GrauData.com/>.

SANavigator Free Trial

A free 30-day trial of Connex's SANavigator software for managing complex storage networks is available for download at <http://www.sanavigator.com/>. SANavigator has an easy-to-use graphical interface designed for enterprise storage management and provides a real-time display of the entire storage area network. Users must complete a registration form prior to downloading the software.

Contact: Connex, Inc., 2040 Fortune Drive, Suite 200, San Jose, California 95131, 408-232-9701, sales@connex.com, <http://www.sanavigator.com/>.

Ascendence Clusters



A new line of Beowulf computational clusters, the Ascendence series, is now available from Atipa Corporation. The clusters come in 4- and 8-node configurations and are available across Intel, AMD and Alpha platforms. Customers can choose from three available high-speed connectivity options, Wulfskit, Myrinet and 100MB Ethernet. For enclosure, clusters are available in 1U, 2U and 3U rackmount cases or full tower cases with hot swappable power supplies.

Contact: Atipa Corporation, 4700 Belleview Suite 300, Kansas City, Missouri 64112, 800-360-4346 (toll free), info@atipa.com, <http://www.atipa.com/>.

Synopsys Design Tool Suite

Synopsys released a full tool suite for high-level design aimed at the growing system on a chip market and other design for testability (DFT) technology. The suite includes the Design Compiler; PrimeTime, a static timing analysis tool; Scirocco, a high performance VHDL simulator; and Module Compiler. The suite provides tools for both design and verification and a simulated graphical tour is available on the Synopsys web site.

Contact: Synopsys, Inc., 700 East Middlefield Road, Mountain View, California, 94043, 800-388-9125 (toll free), <http://www.synopsys.com/>.

Debian Based 2U Servers

VA Linux now offers Debian GNU/Linux software, version 2.2, on its 2200 series 2U Internet servers, along with technical support, services and free updates through the Internet. VA Linux is the first major server vendor to pre-install and support Debian. Automatic software updates are available using **apt-get** from a company-maintained web site that will identify security updates, software dependencies and enhancements.

Contact: VA Linux Systems, Inc., 47071 Bayside Parkway, Fremont, California 94538, 877-VA-LINUX (toll free), info@valinux.com, <http://www.valinux.com/>.

AIT LibraryPro



In collaboration with Enhanced Software Technologies (EST), Overland Data announced that the AIT LibraryPro is certified Linux-compatible. LibraryPro can be scaled up to nine modules for a maximum configuration of 8.55 terabytes native capacity (22.2 TB with compression) and throughput of 388 gigabytes per hour (1.0TB with compression) using Sony AIT-2 drives. EST operates the Linux tape drive certification program to demonstrate compatibility.

Contact: Overland Data, Inc., 8975 Balboa Avenue, San Diego, California
92123-1599, 800-729-8725 (toll free), srichardson@overlanddata.com, <http://www.overlanddata.com/>.

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

[Advanced search](#)

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