



Features

Control Everything from One Place with Synergy by *Chris Schoeneman*

Run the pointer off the edge of the screen...onto a whole different computer? Forget the KVM switch, and use Synergy to interact with all your systems at once.

Scanning with SANE and Other Tools by *Michael J. Hammel*

Here's the software and configuration to make scanning under Linux work.

Linux for a Small Business by *Gary Maxwell*

Can you exchange files with customers and keep track of business books with 100% free software? Small-business owner Gary Maxwell says yes.

The Grand Unified Desktop by *Marco Fioretti*

Applications for a variety of toolkits are coming together in a free best-of-breed desktop. To work together seamlessly, though, they need to follow important new standards.

Indepth

Fixing Photo Contrast with The GIMP by *Eric Jeschke*

If the sky is great while the ground is black, or the ground is right but the sky is washed out, use The GIMP to make the whole photo look properly exposed.

Programming under GNUstep—An Introduction by *Ludovic Marcotte*

Borrow code written for Mac OS X and develop your own applications in Objective-C.

The GNOME 2 Desktop Environment by *Russell Dyer*

GNOME 2 offers better-looking fonts and full-keyboard navigation.

[Hacking Red Hat Kickstart](#) *by Brett Schwarz*

Most of the savings from Linux desktops come from reduced administration costs—like rolling a custom RPM-based load that installs itself.

Embedded

[Driving Me Nuts The USB Serial Driver Layer, Part II](#) *by Greg Kroah-Hartman*

Toolbox

Kernel Korner [The Linux Kernel Cryptographic API](#) *by James Morris*

At the Forge [Content Management](#) *by Reuven M. Lerner*

Cooking with Linux [Sometimes, You Have to Do It Yourself](#) *by Marcel Gagné*

Paranoid Penguin [rsync, Part II](#) *by Mick Bauer*

Columns

Linux for Suits [Subcontinental Smackdown](#) *by Doc Searls*

EOF [Linux Distributions Agree on Standards](#) *by Scott McNeil*

Reviews

[Kylix 3.0 Enterprise \(with C++\)](#) *by Dragan Stancevic*

[Hacker's Delight](#) *by Michael Baxter*

Departments

[Letters](#)

[upFRONT](#)

[From the Editor](#)

[On the Web](#)

[Best of Technical Support](#)

[New Products](#)

[Archive Index](#)

[Advanced search](#)

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

Control Everything from One Place with Synergy

Chris Schoeneman

Issue #108, April 2003

Connect machines to each other and themselves using software instead of KM switchboxes.

What is synergy? The dictionary defines it as an “advantageous conjunction of distinct elements”. The Synergy utility achieves this conjunction by transparently sharing a single keyboard and mouse between two or more computers using a TCP/IP network. Synergy also shares selected text and clipboard selections with full ICCCM (Inter-Client Communication Conventions Manual) and Unicode support. It automatically translates linefeeds between UNIX and Windows formats, enabling cut and paste across systems as easily as within a single system. In addition, it forces screensavers to activate and deactivate in concert. In short, each computer uses its own display(s), and you simply roll the mouse off the edge of one display to jump to another. So, it's almost like having one big desktop spread across multiple computers.

Synergy provides a software replacement for keyboard/mouse (KM) switchboxes. It currently runs on Linux and Windows and has preliminary support for Solaris; any combination of these platforms works. This article describes how to install and configure Synergy between two (or more) Linux boxes. Configuration usually takes only a few minutes.

Building and Installing

First, download the latest stable version of Synergy from SourceForge (sourceforge.net/projects/synergy2). Then follow the usual steps:

```
tar xzf synergy-X.Y.Z.tar.gz
cd synergy-
./configure
make
su -c 'make install'
```

where X.Y.Z is the version number. You can install a prebuilt RPM instead, which is also available from the site. Executables are installed in /usr/local/bin unless you provided a different location to **configure**. Repeat this process on each computer to be connected or simply copy over the binaries (synergyc and synergys).

Configuring the Server

Next, choose the server, the system with the keyboard and mouse physically connected to it. This system requires a Synergy configuration file that names the server, the computers that may connect (the clients) and their virtual screen arrangement. It's a plain-text file with two required and one optional sections. Here's an example configuration file:

```
section: screens
  guava:
  mango:
end
section: links
  guava:
    right = mango
    up    = guava
  mango:
    left  = guava
end
section: aliases
  guava:
    guava.tropical-fruit.org
  mango:
    mango.tropical-fruit.org
end
```

The screens section simply lists the server hostname and the hostnames of all permitted clients. The links section describes the virtual adjacency of the computers. For example, guava lists mango as being located to its right, so when the mouse moves off the right edge of guava it appears at the opposite (i.e., left) edge of mango. Each computer can have at most one of each of the following directions listed: left, right, up and down. A computer also can link to itself; in this example, moving off the top of guava will move the mouse to the bottom of guava.

Links are not automatically symmetric. Making the jump to mango reversible requires that guava is listed as being to the left of mango. This feature becomes more useful with more than two computers. For example, a third computer, banana, could be up from guava and mango, but only one of those could be down from banana.

The third section, aliases, is optional. Clients provide their hostname (or a name specified on the command line) to the server when connecting, so the server can find them in the configuration. Some systems report their fully qualified domain name, others list only their hostname, depending on their network configuration. The aliases section, as it suggests, provides a list of names that

each computer is known as. The above configuration permits mango to connect as mango or mango.tropical-fruit.com. The server also checks the aliases when looking up its own name.

You may have noticed the configuration file doesn't indicate whether guava or mango is the server. That's because doing so isn't necessary. This configuration works as is with either system as the server. For this example, we'll assume guava is the server. Prepare a configuration file for your particular setup using the above example as a template, and save it to `$HOME/.synergy.conf`.

Testing the Client and Server

Now start the Synergy server:

```
synergys -f -1
```

We cover the meaning of the options later on. The server logs some messages to the shell, and if all goes well, it's ready and waiting for connections. Any links from the server to itself in the configuration should work at this point. On guava we could move the mouse off the top edge of the screen, and it would jump to the bottom edge.

With the server running, you're ready to connect a client system. On your other system (mango in our example) start the client with:

```
synergyc -f -1 --no-camp guava
```

replacing guava with the hostname or network address of your server. The client also logs some messages to the shell and either connects to the server or quits with an error. If it connected successfully you can now use the mouse, keyboard and clipboard between the two systems. Test any other clients in the same way.

If the command-line options are invalid or the configuration file has an error, Synergy writes an error message to the shell and quits. If the server or client fails for some other reason, you'll receive a log message prefixed by ERROR or FATAL briefly describing the problem. Space here doesn't permit a complete list of errors, but the message should provide enough information to diagnose the problem.

The command-line options used above indicate that the client and server should run in the foreground, messages should be logged to the shell (-f) and the system should quit when a nonpermanent error occurs (-1). By default, both the client and server run in the background, messages are logged to syslog and the system waits a few seconds then retries after non-permanent errors. The --no-camp option tells the client to quit after the server forcefully

closes a successful connection. Normally the client cleans up then tries connecting again; more on that below. A few other options are available; use `--help` to see a list.

Starting Synergy Automatically

Once you've tested the server and client(s), you'll probably want them to start automatically in the future. Synergy requires an X server, so starting it before the X server starts won't work. The easiest way to start Synergy automatically is to add a line to your `HOME/.xsession` or similar X session startup script. Typically, you'd run the Synergy server from `.xsession` with no arguments and run the client with the server hostname as the only argument. They'd run in the background and quit when the X server quits or restarts.

The problem with this setup is Synergy isn't running during the login screen, which is managed by XDM or one of the equivalents such as GDM or KDM. If you have the necessary permissions, you can reconfigure your display manager to start Synergy when the X server starts. First, copy `HOME/.synergy.conf` to `/etc/synergy.conf` (no leading dot on the latter) so the display manager can find it. Then edit the display manager's Xsetup script; different distributions put this file in different places so you may have to search for it. Near the end of the script but before any call to exit add two lines. You can use either:

```
/usr/bin/killall synergyc  
/usr/local/bin/synergyc guava
```

replacing `guava` with the hostname of your server to start the client, or:

```
/usr/bin/killall synergys  
/usr/local/bin/synergys
```

to start the server. Don't forget to remove any lines in your `.xsession` that try to start Synergy. For security reasons, some display managers (XDM and KDM, but not GDM) grab the keyboard and do not release it until the user logs in. This prevents a Synergy server from sharing the mouse and keyboard until the user logs in. It doesn't prevent a Synergy client from synthesizing mouse and keyboard input, though; log in to the server and then use Synergy to log in to the client.

Without the `--no-camp` option, the client tries connecting to the server every 60 seconds until it succeeds, so the client can start before the server. You can exploit this feature on a laptop: run the client on the laptop all the time. When it's attached to your home network, it'll connect to the Synergy server within 60 seconds. Then you can use the server's keyboard and mouse instead of the laptop's.

Finally, an important note about security. As of this writing, Synergy has no authentication and no encryption safeguards. Because it transmits all mouse and keyboard input, including passwords, do not use Synergy on or across untrusted networks. Future versions of Synergy will address this shortcoming.



email: crs23@bigfoot.com

Chris Schoeneman is a graphics software engineer at Pixar Animation Studios. In addition to Synergy, he's also the author of bzflag. He lives in Berkeley, California, and can be reached at crs@groundhog.pair.com.

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

[Advanced search](#)

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

[Advanced search](#)

Scanning with SANE and Other Tools

Michael J. Hammel

Issue #108, April 2003

Here's how to get started using a scanner from Linux, and a comparison of the features in available scanning software.

A few years ago, Linux users were often confronted by the complex relationship between off-the-shelf video hardware and the XFree86 drivers that made them work. To understand how to configure your new video card, you needed to understand detailed hardware issues, such as which chipset your video card used. While this problem has eased as more video card makers have started to support XFree86 development directly and to provide their own XFree86 drivers, the Linux scanner world still is in that detailed hardware stage.

The Hardware

Scanners can connect to a PC in three basic ways: through a parallel port, through a SCSI bus host adapter or through the newer Universal Serial Bus (USB). It's next to impossible to find off-the-shelf scanners that don't support USB these days, though many also support parallel interfaces as well. SCSI interfaces mostly have been dropped by scanner makers in favor of USB.

USB support in the Linux 2.4 kernel series can be handled either by using the USB kernel scanner driver or by using the libusb library. You can check for the kernel scanner driver by running the following command:

```
lsmod | grep scanner
```

If this command returns anything, you have the kernel scanner driver. If it doesn't, you can load the scanner driver with this command:

```
modprobe scanner
```

This command also will load the USB core module (called `usbcore` when you run `lsmod`) if it's not already loaded. In order for the scanner to work with USB,

the right USB HCI module also must be loaded. For USB 1.0 devices, the module to load is `usb-uhci`. For USB 1.1 it's `usb-ohci`. For 2.0 devices, even when running at lower speeds, it's `usb-ehci`. My USB hardware is USB 1.1, so I have to load the OHCI version:

```
modprobe usb-ohci
```

To use the `libusb` library instead, remove the scanner modules using this command (which should be run as the root user):

```
rmmmod scanner
```

Although support for the kernel scanner driver is rumored to be going away with the forthcoming 2.6 kernel release, it's still common in the current 2.4 kernels. Therefore, for the rest of this article we assume the use of the kernel scanner driver.

You can launch your scanner software using a script that runs the appropriate `modprobe` commands, so you can make sure the scanner driver is already loaded. Alternatively, you can use one of the system start-up scripts, such as the `/etc/rc.local` file commonly used on Red Hat systems, to load the scanner at boot time.

With the scanner driver loaded, next mount the USB filesystem, again as the root user:

```
mount /proc/bus/usb
```

Then you can list the devices on the USB bus:

```
cat /proc/bus/usb/devices
```

This command won't produce any output if you don't have the scanner, USB core and HCI drivers (either `uhci`, `ohci` or `ehci`, as described previously) loaded. The `devices` file is verbose, but what you're looking for are the vendor and product IDs:

```
P: Vendor=04b8 ProdID=011d Rev= 1.00
```

Hang on to these values—you'll need them later if SANE can't find your scanner. If you want to be certain your scanner will be seen by SANE, reload the scanner driver like so:

```
rmmmod scanner  
modprobe scanner vendor=0x4b8 product=0x011d
```

We included the vendor and product IDs this time when we loaded the scanner driver. We also prefixed the IDs with `0x`—this is required if you use the `modprobe` command in this way.

The sane-usb man page gives a more detailed discussion on getting your USB scanner configured. See the Linux USB Project page at www.linux-usb.org for help with general USB configuration and testing.

The Tools of the Trade

Now that we have the basic hardware configuration, we want to make sure the SANE software can access it. The version of SANE used for this article is 1.0.8. SANE software comes in two parts: the back-end driver software and the front-end user interfaces. SANE actually only provides the back-end drivers and a few command-line front ends. X-based graphical front ends, such as XSane and QuiteInsane, are separate projects that make use of the SANE back ends.

Most modern Linux distributions now include a version of the SANE back ends. However, many distributions are providing outdated releases. The SANE web site (www.mostang.com/sane) provides links to current binary distributions in RPM or similar formats for Red Hat, Debian, Mandrake and Slackware.

Once you have the SANE back ends installed, you need to configure the back-end drivers. The first trick is to make sure SANE can find a scanner. The SANE software provides a command-line tool, called sane-find-scanner, that can find any SCSI scanner and most USB scanners. Run this command either with your normal user ID or as root; no command-line options are necessary. The output from this command will be some comments and a line that looks something like this (for USB scanners):

```
sane-find-scanner: found USB scanner
  (vendor = 0x04b8, product = 0x011d)
  at device /dev/usb/scanner0
```

This means SANE can see the scanner using the device `/dev/usb/scanner0`, which is good; thus, we need to configure only this scanner's back-end drivers. However, if you don't get a line like this—if no scanner can be found—when you run **sane-find-scanner** as your normal user, you might have to change the permissions of the device file. You can verify this by running the command as the root user. If **sane-find-scanner** finds the scanner when run as root, the problem is a permissions issue. Assuming you are the only user on your machine, the problem is simple to fix:

```
chown owner.owner /dev/usb/scanner0
chmod 660 /dev/usb/scanner0
```

In this example, *owner* is your user ID and group ID. If you need to share this scanner with other users, you can set up a scanner group instead:

```
chgrp scanner /dev/usb/scanner0
chmod 660 /dev/usb/scanner0
```

These two commands need to be run as root. All users who need access to the scanner must be added to the scanner group. One other note on the device file: you may be tempted to make a symbolic link from `/dev/usb/scanner0` to `/dev/scanner`. Don't. The SCSI back ends use the `/dev/scanner` device name, and linking it to the USB device will confuse the USB back ends.

Now that SANE can see the scanner, it's time to choose its back-end driver. This is the first tricky part. For most Epson scanners you would use the Epson back end. But for the Epson Perfection 1260—an affordable model commonly stocked by most electronics stores—the back end is actually the Plustek driver. For most scanners you can make educated guesses from the SANE web site supported-hardware list. Barring that, you can try to find the vendor and product IDs and match them to the information on the supported-hardware list on the Linux USB Project web site.

SANE Configuration

Now that you have picked the appropriate driver, it's time to configure the back end. If you installed SANE using RPMs or using the default configuration when built from source, SANE's configuration files are located under `/etc/sane.d`. The main configuration file is called `dll.conf`. This file tells SANE which drivers to use. By default, many drivers are enabled. If you have only the one, you can limit this to the driver for your specific scanner. In our example, we've uncommented only the Epson driver, because we're using the Epson KOWA back end instead of the Plustek driver.

Not all back ends support all types of scanners. No matter what type of connection your scanner uses, each back-end configuration file needs to know the name of the device file your scanner will use. Remember, we found the device filename using the `sane-find-scanner` tool. Unfortunately, the format used to define this in the configuration file varies from back end to back end.

The Plustek back end (the default back end for SANE support of the Epson Perfection 1260) includes distinct sections for the USB and parallel port types of scanners. To specify the device file in this configuration file, use the device keyword followed by the name of the device file, as in this example:

```
device /dev/usb/scanner0
```

This entry must go in the appropriate section of the Plustek driver configuration file. However, in the Epson configuration file used by the Epson KOWA back-end driver for the Epson Perfection 1260, the device file is specified using the USB keyword followed by the device filename, as in this example:

```
usb /dev/usb/scanner0
```

Both the Epson and Plustek configuration files provide comments to help in their configuration, and all back ends have their own man pages to provide further configuration assistance. Though many configuration options are provided, the only option really required for all of them is the device file.

To test that your SANE configuration is working, try the following command:

```
scanimage -T
```

If your test fails, you may want to verify once again that the proper USB modules have been installed, that the device can be found by `sane-find-scanner` and that you have the correct device filename in your SANE back-end configuration file. The `scanimage` program's help option also can provide quite a bit of additional information about your scanner's capabilities:

```
scanimage --help
```

Graphical Front Ends

The hardware is configured and ready to run. What you need now is an easy-to-use, front-end graphical interface that lets you preview your scans, select regions to scan from the preview and, perhaps, make color, quality and resolution adjustments. You also need a way to get the scan into The GIMP for further processing.

There are actually three freely available front ends for use with SANE, plus a shareware tool that exists outside of SANE. Let's take a brief look at each before comparing features and quality issues.

XSane

This project has grown up with the SANE Project, side by side. The user interface is based on GTK+, and it includes a GIMP plugin to allow scanning directly from the File®Acquire menu in The GIMP.



Figure 1. The XSane Front End Managing the Epson Perfection 1260

When run as a GIMP plugin, the Viewer window is not used—the scanned image is transferred directly to The GIMP in a Canvas window. Be sure to read the documentation thoroughly to get the most from this interface, including the links to the scanning tips web pages.

The Preview window allows both user-defined and automatic scan regions to be set. Black, gray and white points also can be set in the preview prior to the full-sized scan. The Viewer window provides limited editing.

QuiteInsane

Like XSane, QuiteInsane offers a GIMP plugin. However, this plugin is in early development and may not provide as much stability as XSane's. Beyond this, QuiteInsane offers much of the same functionality as XSane, plus a few extras. QuiteInsane's image viewer allows the user to select regions of the image to work with and permits printing directly from the image viewer.

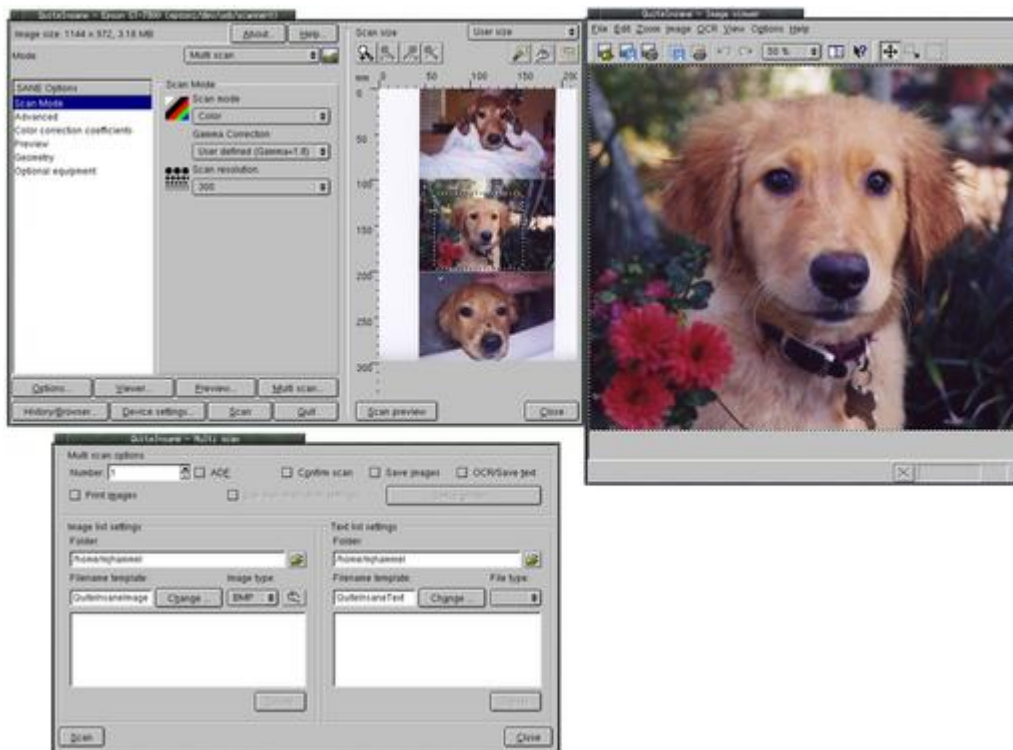


Figure 2. The QuiteInsane Front End Managing the Epson Perfection 1260

QuiteInsane integrates into the KDE desktop, allowing drag-and-drop of images from the Image Viewer into other applications. Although integrating with other applications may not be required, it's a nice feature to have for desktop users.

Image Scan!

The only scanner maker to support the SANE Project actively is Epson. The Epson KOWA Corporation has released their own front end, known as Image Scan!, along with an updated back end that unifies support for all current Epson scanners. This product has the advantage of providing updated drivers for off-the-shelf scanners directly from the manufacturer.



Figure 3. The Image Scan! Front End Managing the Epson Perfection 1260

This front end is less sophisticated than XSane or QuiteInsane, providing fewer features, yet it has a cleaner, less cluttered interface. It also lacks a built-in image viewer, opting to use either The GIMP specifically for image editing or saving scans directly to a file or a printer. There is no built-in help and little on-line documentation.

VueScan

XSane, QuiteInsane and Image Scan! all function as front-end user interfaces to the SANE back-end scanner drivers. However, one other scanner product does not use the SANE back ends at all: VueScan from Hamrick Software. This product is shareware and provides its own set of scanner drivers for a wide variety of scanners.

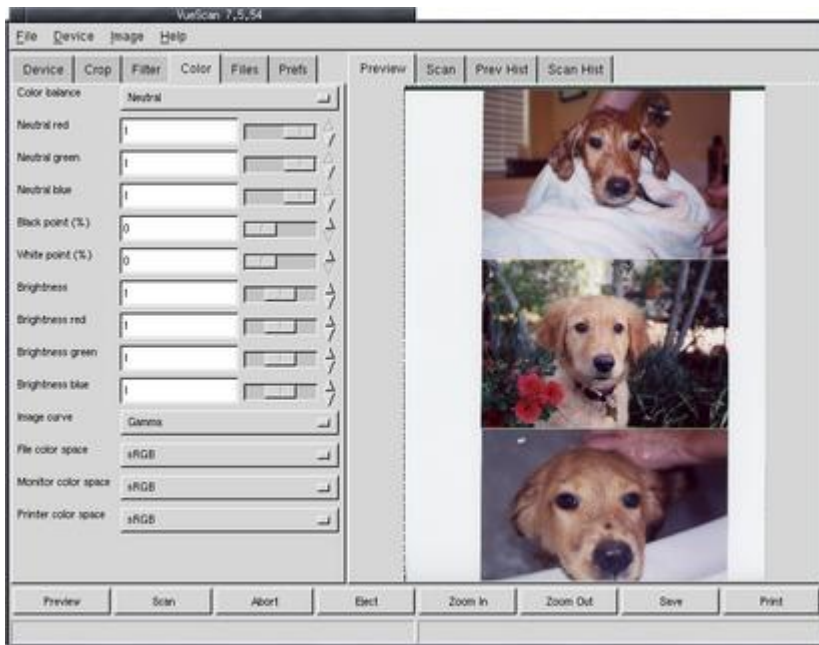


Figure 4. The VueScan Front End Managing the Epson Perfection 1260

VueScan offers many features not provided by the front ends to SANE, such as device calibration, focus and exposure. It does not provide image editing or a GIMP plugin. Scans need to be saved to file first, then opened in an image editor.

Feature Comparisons

Of the four front ends, only one is limited to one vendor—Image Scan!, which supports only Epson scanners. The other three support various vendors. All front ends provide variations on gamma and color channel correction either before or after a scan, or both in some cases. XSane, QuiteInsane and VueScan all provide a built-in image viewer with limited image editing capabilities. Image Scan! scans only to a file or directly into The GIMP.

The areas of largest differentiation in all four front ends are where scans can be sent and how they can be printed. XSane offered the most destinations for scans, including e-mail and FAX. Optical character recognition (OCR) is supported by both XSane and QuiteInsane through the external gocr program.

XSane, QuiteInsane and Image Scan! provide a continuous update to the preview display when a preview scan is in progress. This means you can watch the scan as it happens. The same is true for full scans. VueScan does not provide continuous updates.

Printing is by far the biggest difference in all four. QuiteInsane is the only front end that offers printing from the application, with various print options. XSane can scan directly to the printer, but you can't print a scan from the built-in

image viewer. VueScan offers printing from the image viewer, but the interface is clunky and lacks features.

Documentation for both XSane and QuiteInsane is extensive and fairly well written. HTML documentation is also provided for VueScan but is far less complete. Image Scan! has a minimalist man page.

All of the open-source tools provide GIMP plugins, though VueScan does not. XSane's plugin is the most advanced in stability. QuiteInsane's plugin is feature-rich but is under early development and may not be as stable as users might prefer. Image Scan!'s plugin is functionally equivalent to its standalone version, while XSane and QuiteInsane offer slightly modified versions for their GIMP plugins.

Table 1 is a comparison of the four scanner front ends. Although this table is a good tool for choosing which front end to start with, you would be missing out if you didn't at least try each of them.

Table 1. Scanner Front-End Comparison

	XSane	QuiteSane	Image Scan!	VueScan	
Back End	SANE	SANE	SANE/Epson KOWA (Epson scanners only)	Built-in (supports many makes/models with USB, SCSI & parallel connectors)	
User Interface	Standalone	GTK+	Qt	GTK+	
	GIMP Plugin	GTK+	GTK+ (requires Qt too)	No	
	Other styles	None	Windows, Motif, Platinum, SGI and CDE	None	
	Tooltips	Yes	Yes	No	No
	Preset media types	Yes	No	Yes	Yes
	Help documentation	External browser	Internal browser	Limited man page	Netscape browser
	Linear, Grayscale, Color	Back-end-specific	Back-end-specific	Scanner-specific	Scanner-specific
	Configurable scan resolution	Yes	Yes	Yes	Yes
	Configurable preview resolution	Yes	Yes	No	Yes
	Automatic filename generation	Yes	Yes	No	Yes
	License	Open source—GPL	Open source—GPL	Open source—GPL, LGPL, and Epson KOWA Public License	Shareware
Save File Formats	PNG	Yes	Yes	No	
	JPEG	Yes	Yes	Yes	
	TIFF	Yes	Yes	No	Yes
	PNG/PBM family	Yes	Yes	Yes	No
	XPM/XBM	No	Yes	No	No
	Other	PS, Raw	BMP	No	Raw
Scan To...	GIMP Plugin	Yes	Yes	Yes	No
	Internal Viewer	Yes	Yes	No	Yes, external configurable
	Print	Yes (as Copy)	Yes	Yes	Yes
	File	Yes (as Copy)	Yes	Yes	Yes
	E-mail	Yes	No	No	No
	FAX	Yes	No	No	No
	OCR	Yes	Yes	No	No
	Other	No	Multiple Scans	No	No
Color Correction	RGB Gamma/Intensity levels	Yes	Yes	Yes	Yes
	Color shift (RGB)	Yes	Yes	No	No
	Set white/black/gray points	Yes	No	No	Yes
	Brightness/Contrast	Yes, prescan	Yes, postscan	No	Yes, multiple channels
	Other	No	No	Highlights, shadows, threshold	Color presets, file/monitor/printer color space settings
Image Viewer Features	Selectable regions	No	Yes	No image viewer window	No
	Print from viewer	No	Yes	N/A	Yes
	Undo/Redo	No	Yes	N/A	No
	Filters	Despeckle, Blur	Despeckle, Blur, Invert, Normalize, Oil Painting, Posterize, Sharpen	Unsharp Mask in Preview dialog	Restore Colors, Restore Fading, Grain Reduction, Sharpen
	Rotation/Transformations	Rotate, Mirror	Rotate, Scale, Shear	None	Left, Right, Flip
Preview Features	Autoselect scan region	Yes	Configurable	No	Yes
	Autoselect visible region	Yes	Yes	Manual select-only	Yes
	Zoom preview	Yes	Yes	Requires rescan of selected region	Yes
	Preset scan sizes	Yes, US and European	Yes, European-only	No	Yes, multiple types
	Rotation/Transforms	No	No	No	Left, Right from preview or prescan, Flip, Mirror
	Preview update	Yes	Yes	Yes	No
	Progress bar	No	No	Yes	No

Table 1. Scanner Front-End Comparisons

Quality Comparisons

All scans for XSane and QuiteSane will be essentially the same because of the use of the common SANE back ends. Image Scan! includes a driver that provides the same level of quality as the Epson Windows driver. VueScan's driver is its own. In Figure 5, the two scans compare the Epson and the VueScan driver. Figure 6 shows a close-up of these scans around the eyes. The SANE version has a smoother transition between pixels, and the VueScan provides more detail in the reflection in the left eye.



Figure 5. Comparison: Left—SANE; Right—VueScan. JPEG quality was set to 100 for the SANE version.

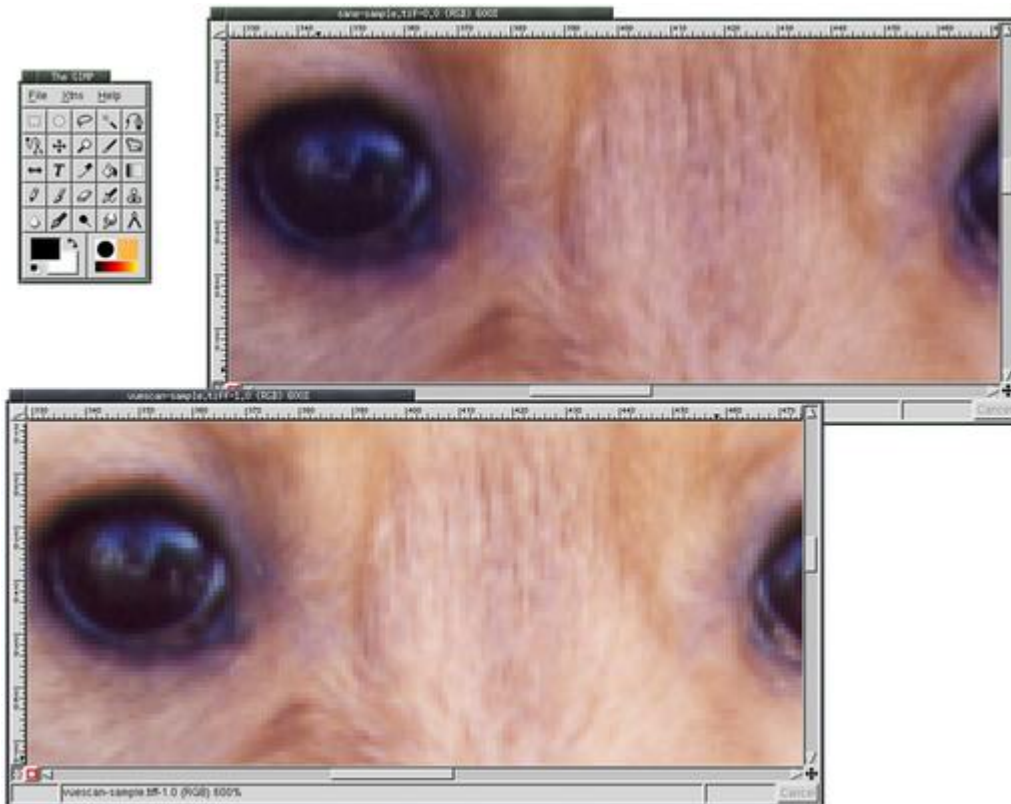


Figure 6. Close-up of the SANE (Epson) and VueScan scans showing variations in quality.

Summary

Each front end has its own unique benefits and drawbacks. XSane's support for scan quality is based on film media options, such as Agfa, Fuji and Kodak film negatives, which give it added ease of use.

Image Scan! is targeted more toward the casual desktop user. As an added bonus it provides a driver straight from Epson that supports commonly available scanners.

VueScan is much faster than the other tools for zooming in on the preview, because it keeps the scan in memory. This makes VueScan preferable to the SANE-based solutions for those who scan large numbers of images.

Only VueScan offers multiple color space support, including sRGB, PAL, NTSC, CIE, Apple, Adobe and others. It was, however, the only one to crash during testing.

Quitelnsane has numerous features that are missing from the other front ends. For example, menubars can be moved, which is a feature this front end inherited from the use of Qt. Other benefits of this tool include a user-modifiable curve graph, similar to The GIMP's Curves tool for adjusting red, green and blue channels; printing directly from the Viewer window with user-configurable scaling; margins; image resolution; and page size options.

However, there are no color-correction presets based on media types, and the available preset scan sizes are in millimeters and non-US standard sizes—no letter or legal options. Despite these few missing features, Quitelnsane has the edge.

Whether you are a professional artist or a casual photographer, there are plenty of Linux-based scanning options. Each front-end user interface offers something different, and back-end drivers are plentiful, with manufacturers such as Epson starting to offer their own versions supporting their specific models. These all combine to put scanning on Linux on par with any desktop.

Resources

Michael J. Hammel (mjhammel@graphics-muse.org) is an author, graphic artist and software developer current working for a storage startup in Houston, Texas. He has spoken at the ALS, LinuxWorld and SXSW conferences and chaired a conference on Linux in Colorado. His web site, The Graphics Muse (www.graphics-muse.com), is an important reference for graphics artists and developers on the Linux platform.

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

[Advanced search](#)

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

[Advanced search](#)

Linux for a Small Business

Gary Maxwell

Issue #108, April 2003

If you're self-employed or run a small shop, here are some office applications you should be using.

In a recent article, an author asked a question: "Is Linux ready for mission-critical production environments in the enterprise?" His answer was an unqualified yes, which shouldn't be too surprising. Anyone who has ever heard of Linux knows of its legendary strengths in the server space. A better question—at least from a writer's perspective—would be: "Is Linux ready to handle the day-to-day demands of a small, home-based writing or consulting business that doesn't have an IT staff?"

Improved hardware detection and ease-of-use distributions such as Mandrake, SuSE and Red Hat have helped Linux make its way onto more desktops than ever before. Add to this the constant refinement of Linux applications, such as office suites, e-mail clients, contact managers, fax clients, web browsers and financial software, and you have a system that places viable alternatives into the hands of every writer and consultant.

I made this discovery almost a year ago. In the spring of 2002, I took inventory of the software I used to run my commercial writing business and found that Linux provided me with a viable, free alternative to every proprietary application I used. This fact, combined with the stability and security of Linux, made the decision to move my small business to the platform an easy one. And I haven't looked back.

Maybe you're a writer or consultant who is seriously thinking about alternatives, as I once did. Or perhaps you've heard of Linux but aren't sure exactly which applications to use. In this article, I provide an overview of four programs found in any Linux distribution that a writer or consultant can use to run a business. Space does not allow me to cover every feature of each application. But I hope to cover enough of the basics so the reader has a good

feel for what these applications can do and how they can be used in a small business.

The Linux applications I'm about to discuss may not have all of the features of their proprietary counterparts. That is, applications provided under Linux have the same functionality but not necessarily the same bells and whistles. We shouldn't be concerned with bells and whistles here, but rather with using tools that allow us to get our work done at a relatively low cost and in a stable, secure environment. If that's what you're looking for, read on.

Tools of the Trade

A small writing or consulting business has minimum requirements where software is concerned. I suppose that's one of the perks of this type of endeavor. The following applications are typically the most commonly used in the operation of a small, home-based writing or consulting business: an office suite, a contact manager/e-mail client, a web browser and financial software. Anything more is gingerbread.

OpenOffice.org

OpenOffice.org is my office suite of choice. It is 100% open source, runs on several platforms and is freely available at www.openoffice.org. One of the OpenOffice.org suite's greatest strengths is its use of XML-based file formats. XML is a structured metalanguage that easily can encapsulate files for distribution between computer systems that otherwise would be incompatible. So, the longevity of your data is guaranteed.

The OpenOffice.org suite consists of four different applications: OpenOffice.org Writer, OpenOffice.org Calc, OpenOffice.org Impress and OpenOffice.org Draw.

OpenOffice.org's Writer is the word processor in the suite. Its interface has a familiar look and feel; it's similar to Microsoft Word and Sun's StarOffice Writer (Figure 1).

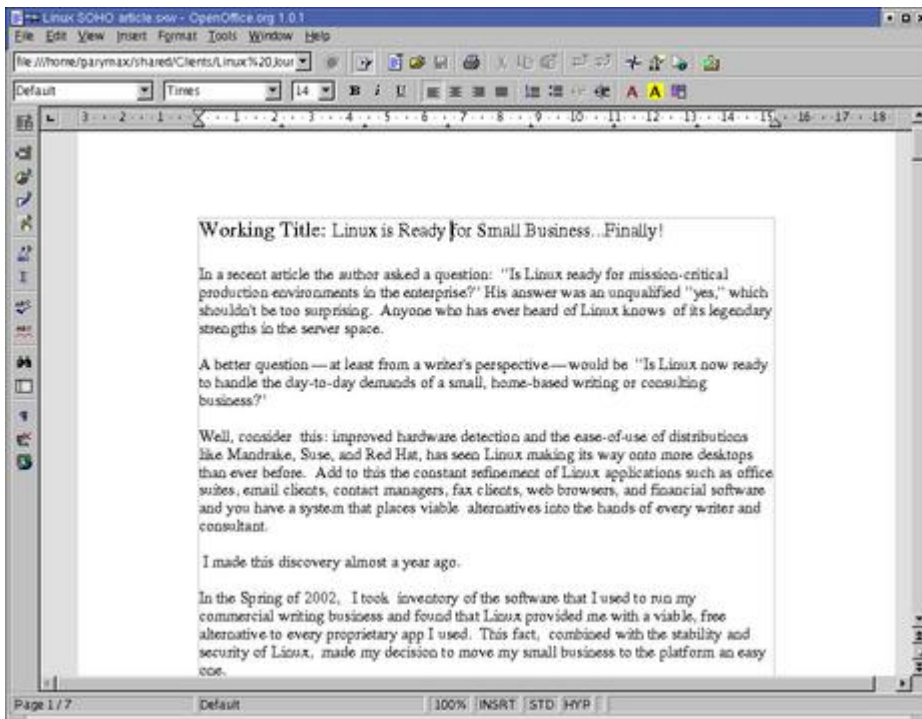


Figure 1. Doing My Assignment

As a commercial writer I use only about 10-15% of a word processor's power. My clients do most of the graphic design and formatting. From business letters and proposals to articles and books, OpenOffice.org Writer is more than enough to get my work done.

The Writer interface is intuitive and laid out in a manner conducive to finding what I need when I need it. It has everything necessary for today's established freelance writer/consultant and then some, and it saves data in several file formats, including Microsoft Word 6, Microsoft Word 95, Microsoft Word 97/2000/XP, rich text format (RTF) and StarWriter 5 (an early StarOffice file format). It even exports PDFs.

One caveat: if you use a lot of tables or special formatting, some of it may be lost or garbled when exporting to Microsoft Word or another office suite. A good rule of thumb is the simpler your layout, the better its compatibility with other word processors.

Because the needs of a writer are few, I hardly use the other programs in the OpenOffice.org suite. But as far as spreadsheets go, OpenOffice.org's Calc has all of the ordinary features one would expect in a spreadsheet, including autosum, autoformat, graphs and many other functions. It saves work in Microsoft Excel, StarCalc, the Data Interchange Format and, of course, in its own format (Figure 2).

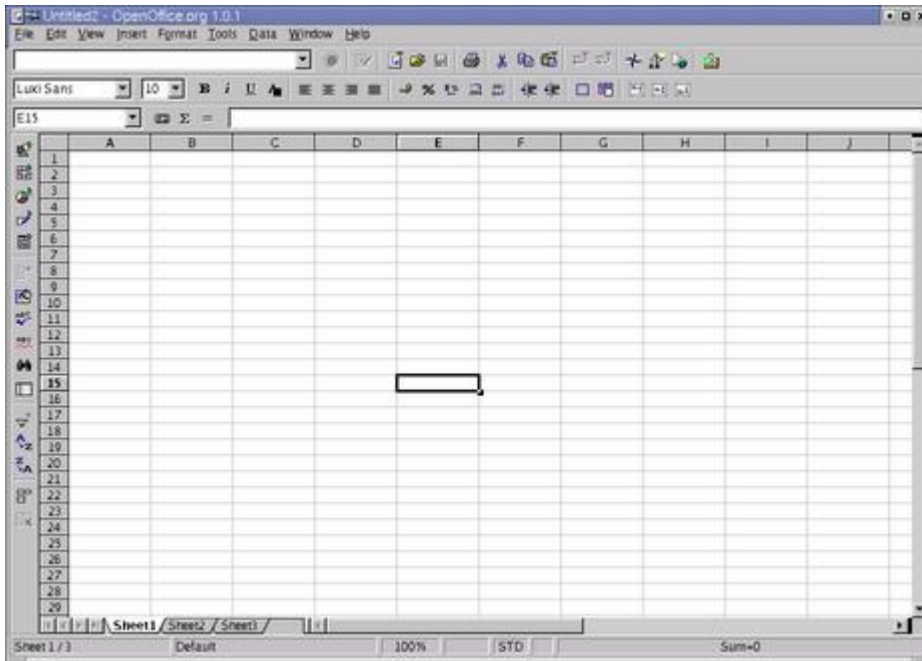


Figure 2. The OpenOffice.org Calc Spreadsheet

OpenOffice.org's Impress, a presentation program, has all of the basic features of Microsoft PowerPoint, but it lacks templates. This lack of templates makes more work for you when designing presentations. Impress reads and writes PowerPoint, StarImpress and its own file format (Figure 3).

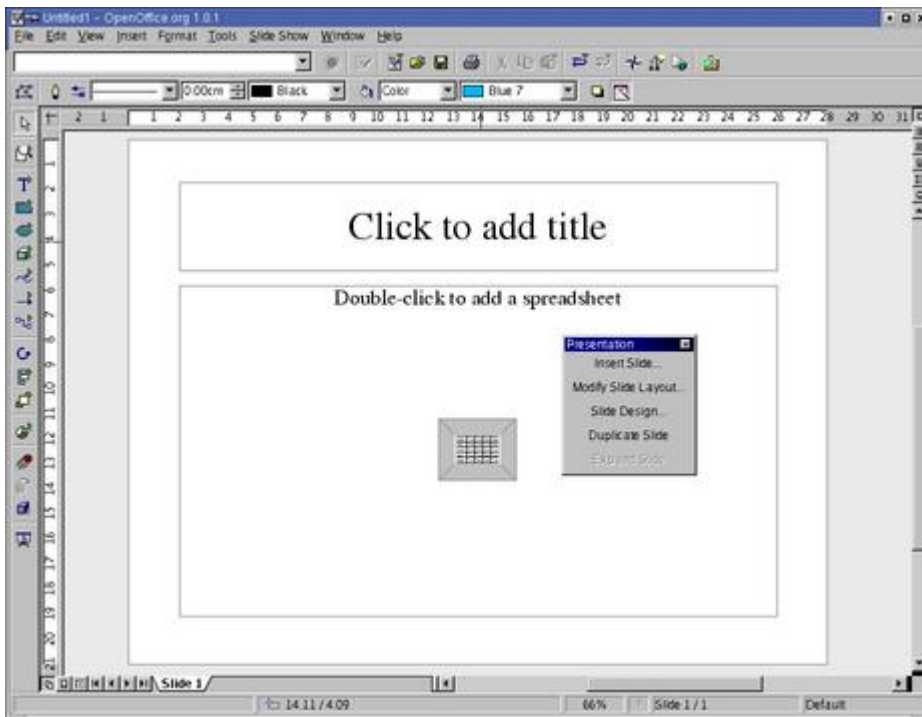


Figure 3. Building a Presentation

The drawing application, Draw, has all of the usual graphic features, including the ability to read the most common types of graphic files. It saves to OpenOffice.org's Draw format and the StarDraw format, and it is good for basic needs (Figure 4).

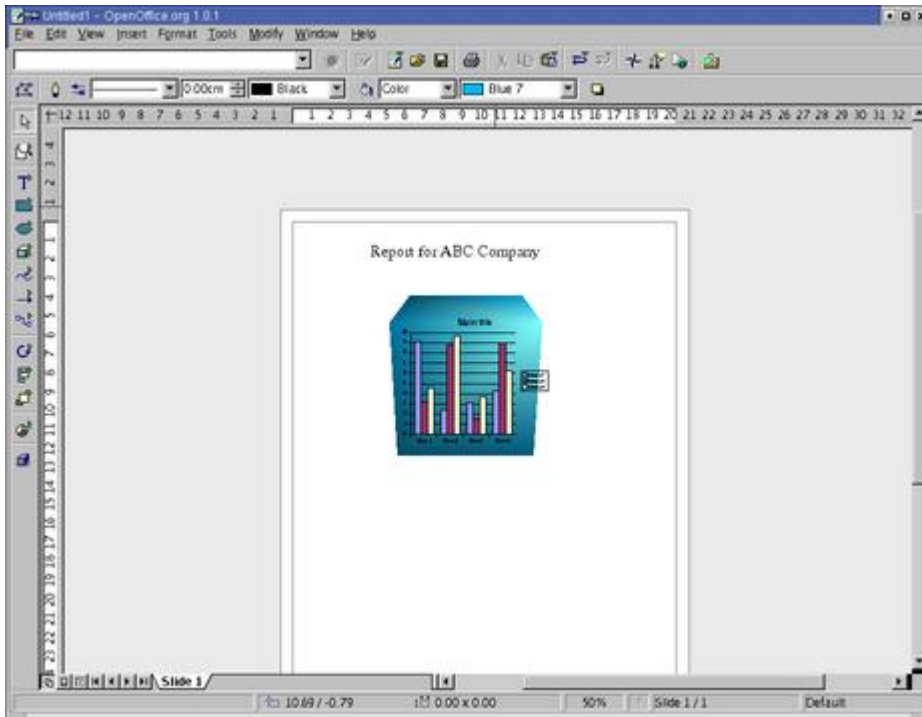


Figure 4. Drawing a Graph

Contact Manager/E-Mail Client

Although Linux has many available options for both contact managers and e-mail, I prefer to combine them. Doing so makes things more convenient—everything is right there, so I don't have to open two different applications. Ximian Evolution fills this need quite nicely. It is at once an e-mail client and a contact manager/task scheduler, and it provides the writer/consultant with a one-stop information resource. Upon starting the program for the first time, you will be taken to the summary screen (Figure 5).

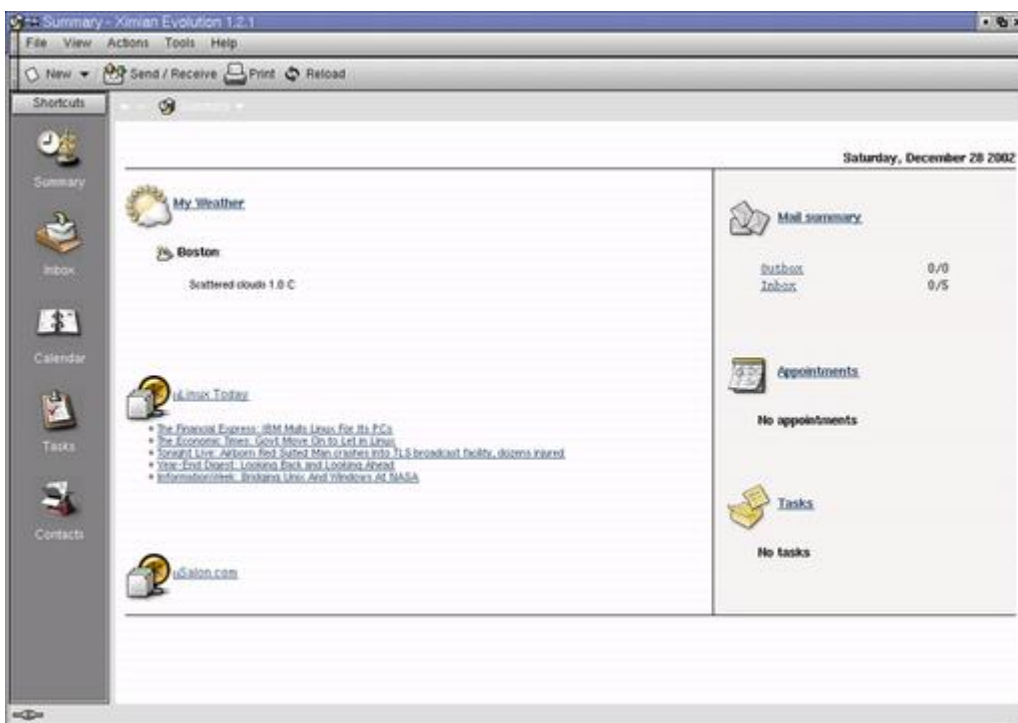


Figure 5. Starting Ximian Evolution

Along with weather information and a listing of recent articles about Linux, Evolution shows how many messages are in your inbox and outbox. It also shows any pending tasks or appointments for the day. Down the left-hand side of the Summary Window are the different areas within Evolution that allow you to set up appointments, schedule tasks and send and receive e-mail. Clicking the Inbox icon will take you to Evolution's e-mail client (Figure 6).

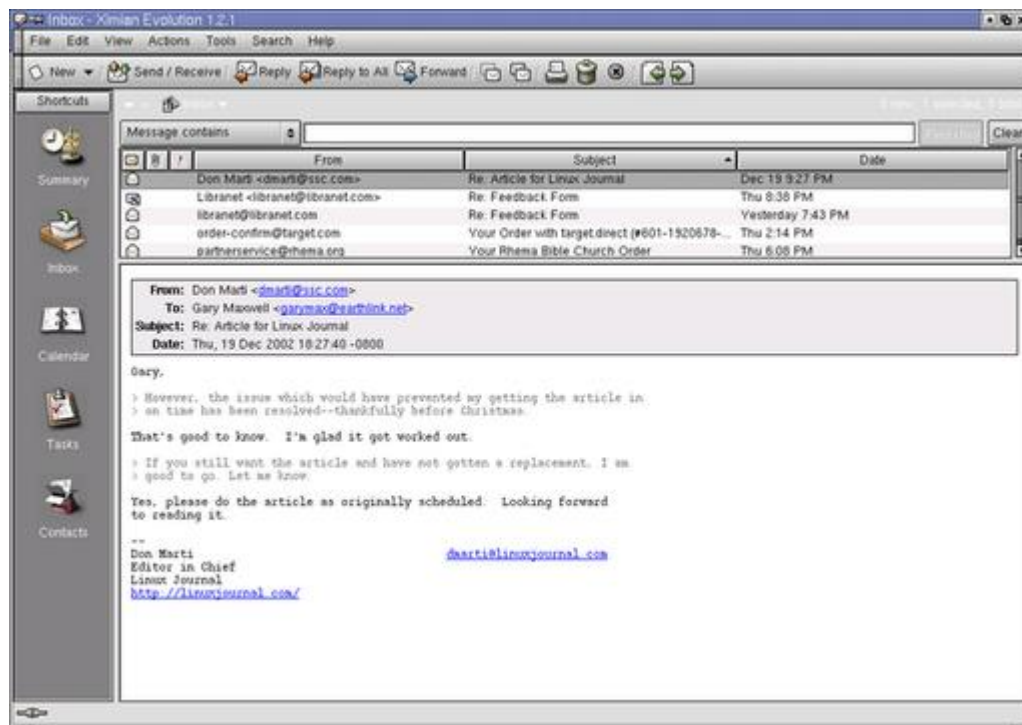


Figure 6. Checking the Inbox

Evolution has a look and feel similar to Microsoft's Outlook program. All of the usual e-mail buttons and services are listed across the top of the page. One nice feature with the release of Evolution 1.2.1 is the New button. With a click, you can create a new mail message or contact without having to navigate to that specific area of Evolution.

With Evolution's contact client, contacts are easily created and managed (Figure 7). I can enter the usual name, address, phone number and so on, but I also can click on the details panel and *viola!* I have another screen to record additional information. And, the more information I have on clients, the better my ability to communicate with them. In case you're wondering, the collaboration panel allows you to record the URLs for those clients who publish their calendar information on the Internet. It's another example of how Evolution's developers have put a lot of thought into the components of this program.

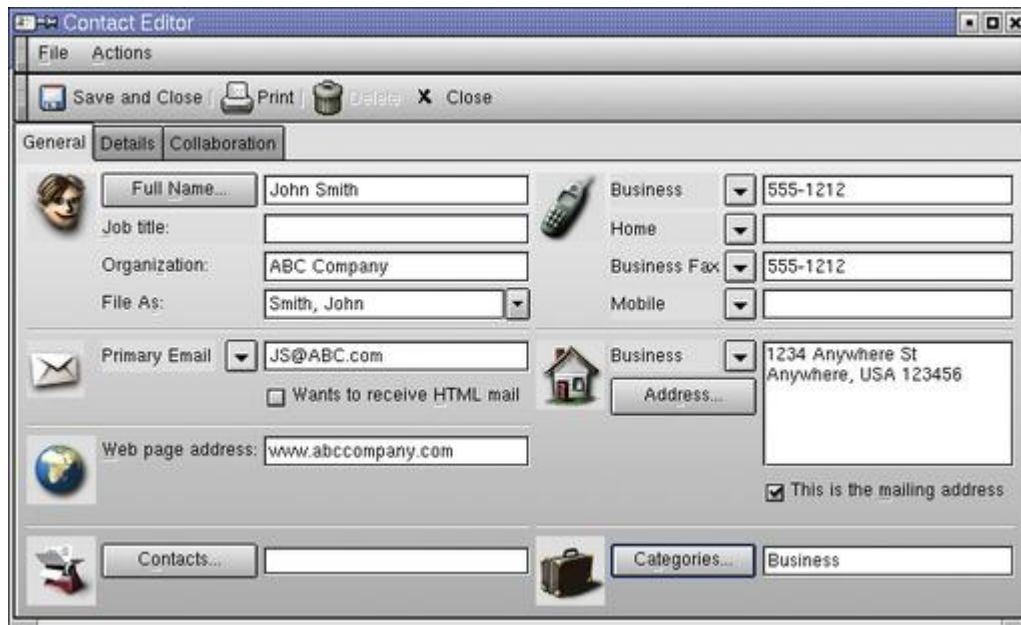


Figure 7. A Contact in Evolution

Once I build a list of contacts, I can search the list based on any number of different search criteria: alphabetical order, e-mail address or category (Figure 8). You even can create your own search criteria using the Advanced selection. You then can file contacts under any category you choose and search for them accordingly. For me, this makes prospecting a snap.



Figure 8. Classifying a Contact

The Calendar client (Figure 9) is pretty straightforward, and you can set up the calendar window to suit your tastes. The window areas can be changed simply by dragging the borders to the desired size. Once you've navigated to the desired day, double-click on the appointment time and a separate window opens up (Figure 10). You then can fill in the details of your appointment and store it for later viewing. You also can set up your calendar to remind you several minutes, hours or days in advance of an appointment.

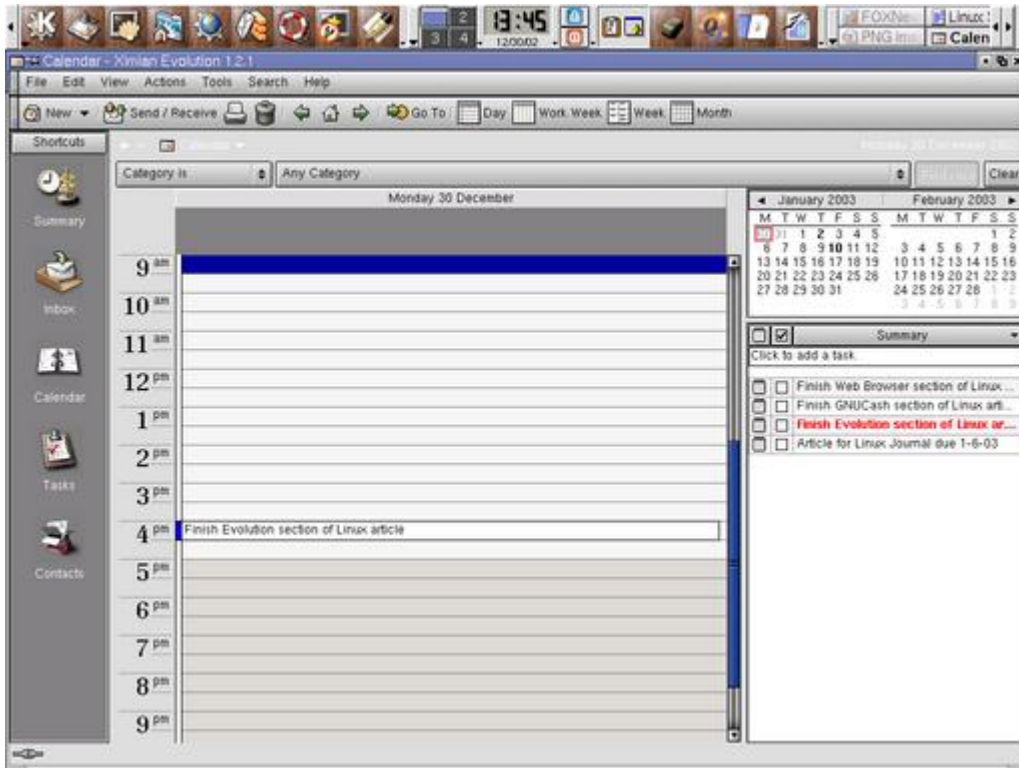


Figure 9. Evolution Calendar

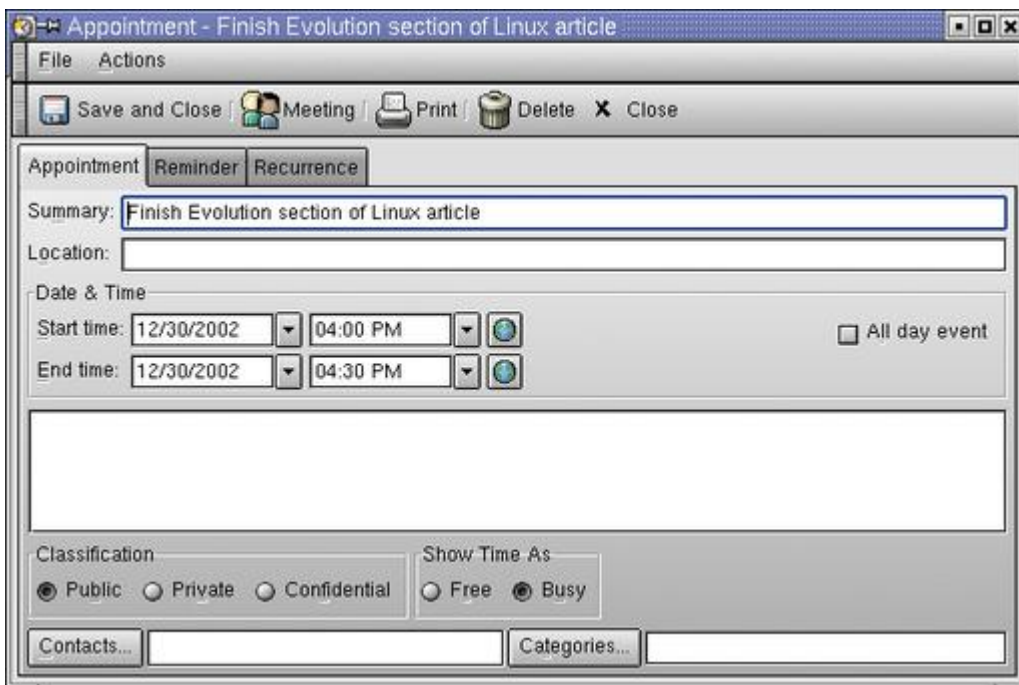


Figure 10. Appointment Specifics

The Task Scheduler works in a similar fashion. Navigate to the New button and choose task. A window opens up that allows you to type in the details of your task. The tasks are stored in the “task” area of Evolution and can be displayed in order of time and date, allowing you to see whether you are on track in a project or group of tasks. Also, once a task is completed, you can check it off and know exactly what you have finished as well as what else you have left to do.

Several more features are available that you can explore for yourself. Like Linux, Evolution allows you to accomplish the same task in several different ways.

A Web Browser

By far, two of the most popular web browsers are Mozilla (Figure 11)--based upon the same rendering engine as its commercial counterpart, Netscape Navigator—and Galeon (Figure 12), a web browser that sports many of the same features as Mozilla and is part of the GNOME desktop.



Figure 11. Mozilla



Figure 12. Galeon

In addition to these two browsers, Linux also offers Konqueror, the KDE-based file manager/web browser. Most Linux browsers support 128-bit encryption for secure transactions.

Speaking of transactions, it's usually a good idea to have more than one web browser available. The reason? Web site access. At first, you think a browser is a browser; other than the look and feel, they all do the same thing. Even though it is popular, Mozilla lacks a feature that Konqueror has: changeable user agents.

When you connect to a web site, your browser identifies itself to the server, offering its name, version number and the system on which it is running. Though most web sites are browser agnostic, some will give you a hard time if you do not have a certain kind of browser.

For instance, every time I tried to log in to my telephone company's web site I always had problems. The site never let me get past a certain point. After scratching my head for a while, I remembered hearing reports of some web sites not supporting browsers other than Microsoft's Internet Explorer.

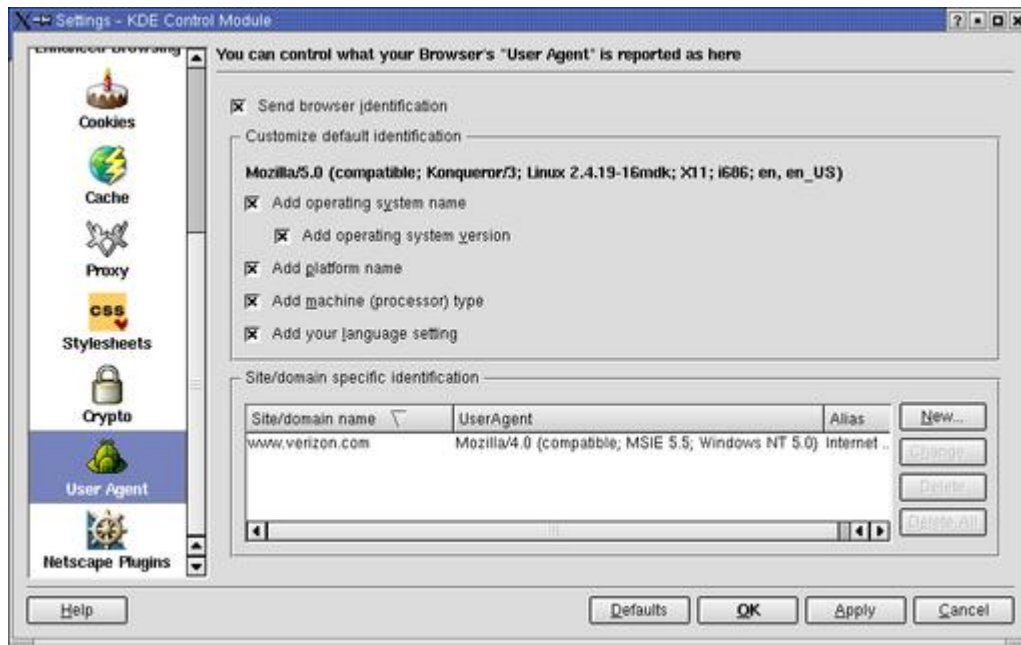


Figure 13. Konqueror's User Agent Feature

This is where the User Agent comes in (Figure 13). I simply changed my web browser's identity to Internet Explorer 5.0, and my web site transaction went through without a hitch. You won't run into this situation too often, but if you do, try changing the user agent and see what happens. Again, this demonstrates how Linux offers the user many different ways of accomplishing the same task in addition to tools for overcoming proprietary obstacles.

Financial Software

Once you earn your money, you need to keep track of it. Financial software usually fills the need. One of the best-kept secrets of Linux is the open-source accounting project GnuCash (Figure 14).

Accounts	Account Name	Description	Total
Account Summary	Imbalance-USD		\$0.00
	Assets	Assets	\$6,601.19
	Current Assets	Current Assets	\$6,601.19
	Cash in Wallet	Cash in Wallet	\$0.00
	ABC Bank Savings		\$6,601.19
	Liabilities	Liabilities	\$0.00
	Accounts Payable	Accounts Payable	\$0.00
	Credit Card	Credit Card	\$0.00
	ABC Printers		\$0.00
	Income	Income	\$3,700.00
	Bonus	Bonus	\$0.00
	Gifts Received	Gifts Received	\$0.00
	Interest Income	Interest Income	\$0.00
	Other Income	Other Income	\$0.00
	Writing Income		\$3,700.00
	Expenses	Expenses	\$99.81
	Adjustment	Adjustment	\$0.00
	Auto	Auto	\$0.00
	Bank Service Charge	Bank Service Charge	\$0.00
	Books	Books	\$99.81
	Cable	Cable	\$0.00
	Charity	Charity	\$0.00
	Clothes	Clothes	\$0.00
	Computer	Computer	\$0.00
	Dining	Dining	\$0.00
	Education	Education	\$0.00
	Entertainment	Entertainment	\$0.00
	Gifts	Gifts	\$0.00
	Groceries	Groceries	\$0.00
	Hobbies	Hobbies	\$0.00
	Insurance	Insurance	\$0.00
	Laundry/Dry Cleaning	Laundry/Dry Cleaning	\$0.00
	Medical Expenses	Medical Expenses	\$0.00
	Miscellaneous	Miscellaneous	\$0.00

Figure 14. GnuCash Tracks Your Dollars

GnuCash is a robust, easy-to-use accounting system that makes balancing your personal books a snap. And GnuCash's Setup Wizard allows you to set up multiple accounts with different opening balances, simply by answering a few questions and making a few selections (Figure 15).



Figure 15. GnuCash Setup

Although GnuCash sports features of its proprietary counterparts, such as transaction auto-completion and check number auto-increment, it uses double-entry accounting, like professional accountants and enterprise accounting software.

The idea behind a double-entry system is that there is a debit from an account and a credit to an account for each transaction. This way, you know where the money came from and where it went. For instance, if I want to pay for some business-related books, I would debit my savings account and credit an expense account for books (Figure 16). This is an invaluable tool when you need to find out where all of your money went.

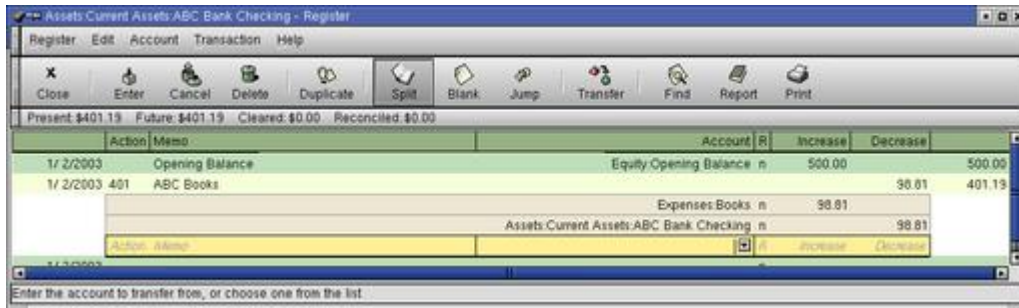


Figure 16. Where Did the Money Go?

What is really nice about GnuCash is you can customize it to suit your personal financial situation. A list of common accounts comes ready-made—expenses, income, stocks, investments—each with its own sub-account structure. However, you also can add your own accounts and delete those you don't use (Figure 17).

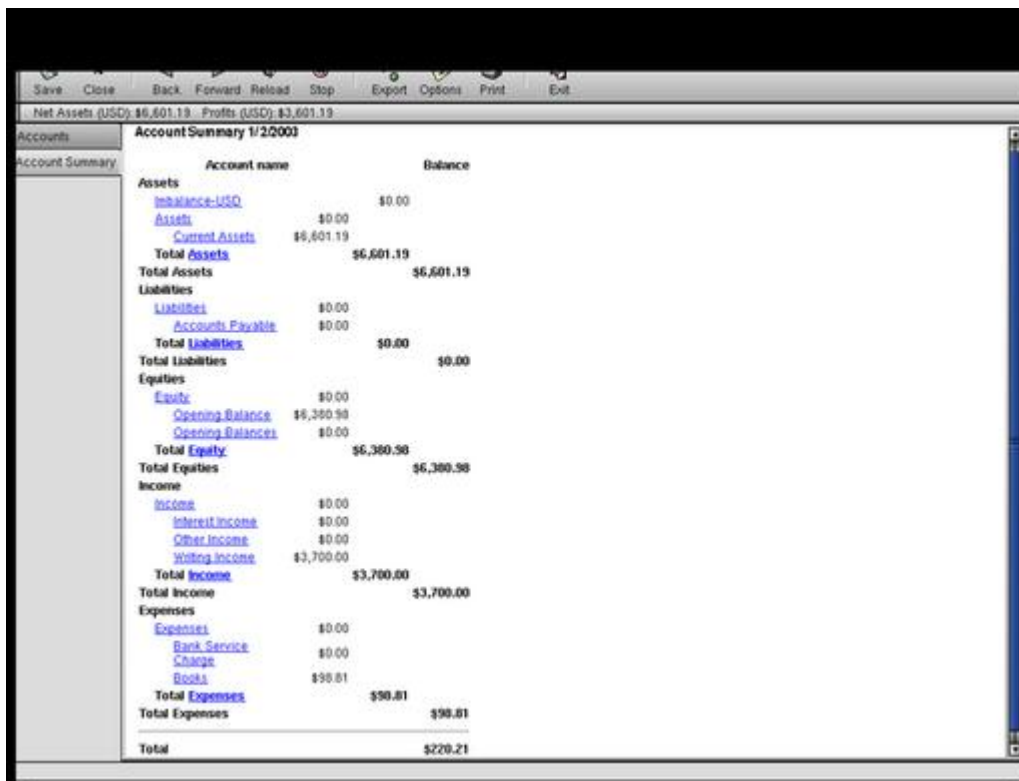


Figure 17. Customizing the Accounts

Another nice feature of GnuCash is the ability to split transactions, so you can include several debit or credit accounts in a transaction. This allows you to fine-

tune a transaction and show every account involved in a specific financial activity.

Lastly, GnuCash has several graphical reports that show at a glance your net worth, your income, your expenses, profit-loss and so on. This alone allows you to manage your finances with greater precision.

Many more features—too numerous to mention here—come with GnuCash, but suffice it to say that for a one-person shop, GnuCash has all of the features you need and then some.

At the time of this writing, GnuCash is at release 1.6.7 and is designed to be used for personal finances. No business features, such as customer and vendor tracking or invoicing and bill payment, exist. But you can customize GnuCash to balance the books for a small business with a little tweaking, as mentioned previously. With the release of GnuCash 1.8, small-business accounting features will be available, including invoicing and bill payment.

Conclusion

If you are currently using a proprietary operating system and proprietary applications and have decided to make the switch to the security, stability and freedom of Linux, I welcome you, and I consider my mission accomplished.

After spending nearly 15 years with a Fortune 100 company, **Gary Maxwell** started a second career as a commercial writer. A Linux enthusiast, Gary operates his business completely with Linux and open-source software. He specializes in advertising copywriting and corporate communications. For questions or comments, he can be reached at gary@garymaxwell.com.

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

[Advanced search](#)

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

[Advanced search](#)

The Grand Unified Desktop

Marco Fioretti

Issue #108, April 2003

How developers and distributions can make diverse software work together better.

In December 2002, I wrote an article for the *Linux Journal* web site (www.linuxjournal.com/article/6476) about the new Bluecurve desktop for Red Hat 8.0, which includes programs from both GNOME and KDE. I received a lot of feedback, including many suggestions on how to write desktop applications that would cooperate with applications written using other toolkits and work well in any desktop environment.

The Problem

An integrated desktop is one whose components work together, wherever they come from, and one that never forces the end user to do the same thing many times. The current default solutions, like GNOME and KDE, often present limits if one tries to mix pieces.

Communication between graphical clients, from simple drag-and-drop to the handling of icons, menus, metadata such as URLs and, in general, window-level interactions, work out of the box only for some subsets of clients and window managers. The same applies to centrally managed and good-looking fonts.

And what about localization? How many programs are still a pain to use with a non-English language or keyboard?

KDE and GNOME are two collections of applications that seem to solve many of these problems. But are toolkit-centric, all-or-nothing approaches the easiest to maintain? What if somebody creates an even better toolkit? What about applications from other toolkits or the whole best-of-breed philosophy so natural among free software users?

In other words, what is the best way to develop that killer application for your favorite desktop, other people's favorite desktops and the desktop of the future?

The Solutions

Re-inventing the wheel may be silly, but it's much less silly than inventing a wheel that needs a new road. Consequently, even if there already are 20 chat clients for Linux, code another one if it feels good; however, make sure that it interoperates with what already exists. Start with common sense, and use the right free standards and protocols. Afterward, check that whatever toolkit or library you choose respects them too. Many top developers already are fully aware that this is the winning approach in the long term. GNOME developer Havoc Pennington wrote, "Interactions between applications and the runtime environment really need to take place via documented, toolkit-independent protocols and file formats." Let's look at how to do this in practice.

File Formats

The most interesting thing in the office formats field, but also for many other types of structured data storage, probably is the work of the Oasis Group. The Oasis file formats are being developed starting with the OpenOffice.org ones, so OpenOffice.org developers will have an easier time. But every program, from Emacs to KOffice to the next word processor, can use them.

Along these lines, there's no need to create yet another bookmark format. XBEL (XML Bookmark Exchange Language) is a bookmark interchange format that Galeon and Konqueror already use.

The idea of an interchange format is valid for other configuration files too. UNIX and Linux still are following the toolbox approach: no mammoths, but a lot of little pieces, each doing one thing well. One practical consequence of this is that all the applications doing the same thing will need to know and store the same set of parameters, or very similar ones.

Sticking to the chat client example, the really silly thing would be to have one configuration file for Qt_chat_app, one for GTK_chat_app, one for MyToolkit_chat_app and so on. There is a fundamental difference between a file and the way it is created or updated. Using different editing methods are okay (vi, a control panel, etc.), but in the long term, what really matters is having a single ~/.chatrc file with every existing chat client using it and not complaining if the settings changed from another chat client. Usenet newsreaders already use the standard .newsrc file. Until a standard emerges for your category of application, an acceptable compromise is to have your new program load the settings of other similar clients when it first starts and use them as defaults.

Guessing the MIME type of a file from its name is an action performed looking at a MIME database. This can be unified too, so that every desktop and program is guaranteed to read the same data; freedesktop.org has a draft specification of how to do this.

Graphical Interfaces

Many people say that X is showing its age and point to ambitious, far-reaching projects like Fresco. Even without aiming so high, however, a lot can be done to make sure that any mix of clients and window managers behave properly. A starting point for approaching this kind of problem is the freedesktop.org site, especially its standards section.

Remember what we just said about applications of the same type sharing the same set of configuration parameters? We were referring to the core functionality, but the same approach can be used with all GUI-level settings, in all the components of your desktop. Things like background color or the double-click timeout are needed by all programs and still can be configured once and for all by each user, regardless of the particular mix of toolkits he or she needs. The X Resource Manager was not designed to be changed interactively and merges what is written in `~/.Xdefaults` with data from other sources, so it is not the right back end for a GUI configuration tool. Go for the Xsetting specification instead, which is being designed to solve this and other problems.

Let's go one step further. What comes after the internal configuration of each class of applications and the way they appear and react to local user actions? The graphical interaction with other applications and the window manager.

Dragging and dropping text from any window to another one already is possible, theoretically at least, if all interested parties follow the XDND protocol. This currently is supported not only by GTK+ and Qt, but also by script-oriented toolkits like Tk and Perl/Tk and by relative outsiders like FOX.

Figure 1 shows how XDND works. When an application must send something to another one, it issues the proper toolkit call. The toolkit sends everything, through the XDND protocol, to the other toolkit, and the answer is passed back to the program in the same way.

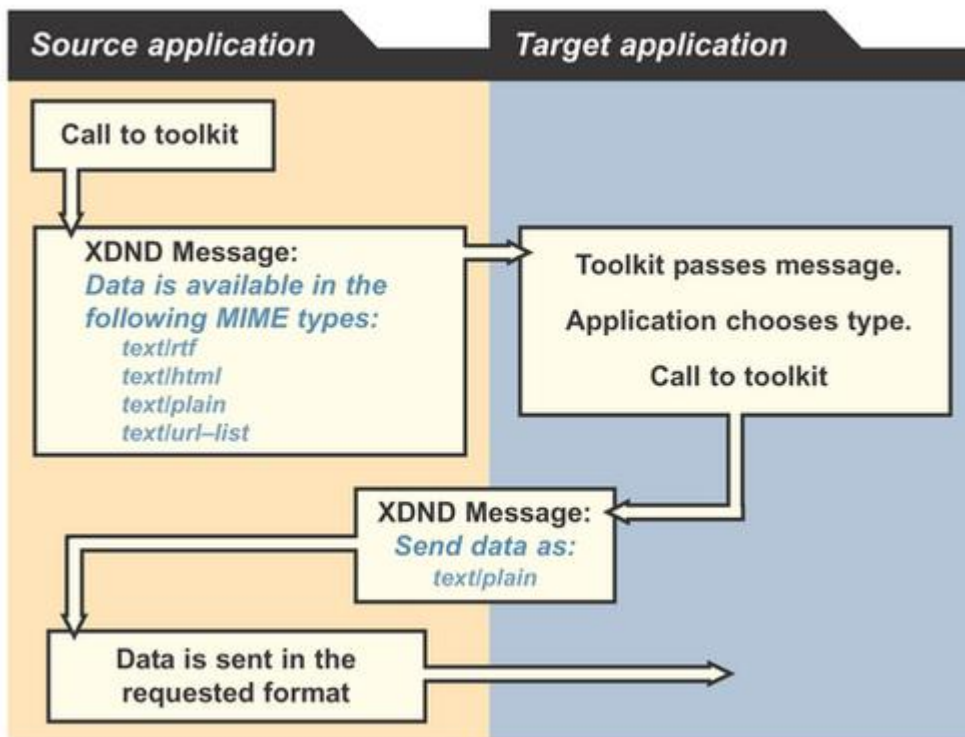


Figure 1. XDND, the Drag-and-Drop Protocol for X

The source must declare all the MIME types it supports to its toolkit. These can be text in many formats, images or filenames. The target must declare which formats it recognizes inside that list. In this way, formatted text can be passed between two word processors without losing font, color and so on. At the same time, passing a paragraph from, say, KOffice to vi will lose formatting but keep the text intact.

When drag-and-drop doesn't work, the problem is almost always inside the application's XS, which didn't declare all the MIME-types, used only custom ones or misused either the toolkit or the protocol in some similar way. Practically speaking, it means that bug reports must be filed against the *applications*, not the toolkits or XDND.

Other interactions between X clients, or between clients and the window manager, must follow the ICCCM (Inter-Client Communication Conventions Manual) and the Extended Window Manager Hints, EWMH for short, formerly known as the NetWM specification. It should be noted that even the good old X clipboard is fully described in the ICCCM. Detailed explanations of how to handle this are provided by Keith Packard and Jamie Zawinski.

The second window interaction standard works on top of ICCCM. It deals with all the window management features not specified in its predecessor, because they started to appear after it. EWMH originally was meant to be a replacement of what was then the GNOME window manager specification, but it is designed

correctly in that it can be implemented and supported in any desktop environment. GNOME 2 and KDE 3 both support EWMH, so any application conforming to it can be expected to play nicely inside both environments or with any of their components. Developers willing to have a more precise, yet quick feel of what EWMH is supposed to do may want to look at KDE's netwm.h.

Fonts and Encoding

Have we covered all that is needed to build a Grand Unified Desktop? Certainly not. Consider Red Hat 8.0. Almost everybody, including those who hate the idea of a GNOME/KDE hybrid, agreed that at least the fonts look much better than before. The way to duplicate this particular bit of magic on your distribution of choice is well known.

First, the improvement is due to anti-aliasing, which makes characters smoother by adding pixels in the proper places. Figures 2 and 3 show the same text in a standard xterm and in an anti-aliased one.

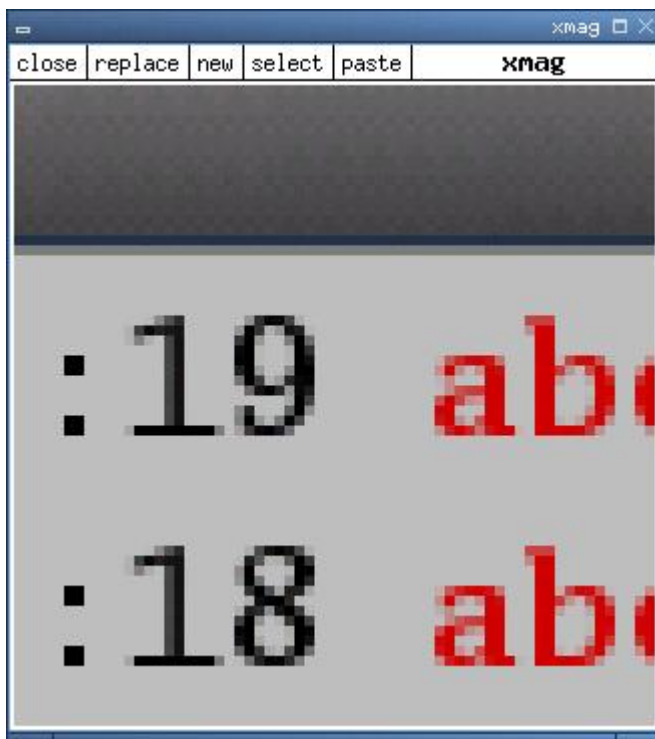


Figure 2. Text in a Standard Xterm



Figure 3. Anti-Aliased Text

Second, Red Hat 8.0 is the first mainstream distribution to have used as much UTF-8 encoding as possible (more on this later) and the new client-side font-management system implemented by the xft2 and fontconfig libraries.

Basically, fontconfig can figure out what fonts are available in the system and which is best for each document. Once this is known, xft2 can tell the X server what to draw. Both libraries need to interact with FreeType or an equivalent rasterizing engine. In other words, font detection and rendering have been cleanly separated but packaged in a way that any client can embed. This means that:

- Eventually, font servers should not be needed anymore.
- Installing new fonts, even without the root password, is much easier.
- Any application using fontconfig reads all font configurations from the same XML file(s), which can be edited by any front end.
- Font management (in any application) can now proceed at the speed of these two libraries, not at the speed of X itself.
- Because fontconfig doesn't require xft2 or any other X-related element, it can be embedded in anything that deals with fonts, including print drivers and libraries. libgnomeprint22 is doing exactly that.

Another nice thing about the xft2/fontconfig system is that it is not an all-or-nothing deal. It can coexist peacefully with traditional font servers, which still may be needed to assist older applications. This is what happens with Red Hat 8.0. And, xft2 can talk to both old and new XFree86 servers. The first ones

receive long sequences of low-level drawing instructions. The others, compiled with the Render extension, receive faster, more sophisticated commands.

When drawing nice symbols is only half of the work and an application also needs advanced text layout, the tool of choice is Pango, which uses fontconfig. Pango is another tool born inside a more or less monolithic desktop, GNOME, but currently is developed to be usable in any other environment.

One last word about fonts—nicer drawings are cool, but if they were only used for digits and the English alphabet, they wouldn't really be worth the effort, would they? Moving to Eastern European, African or Asian languages, ASCII can't even handle "Hello, World!"

For developers, this means any new application must be coded to deal with multilanguage encodings from day one, and all existing programs must be at least checked to guarantee that they still will work. This is not merely a suggestion. If you think you are safe because you only use ASCII and only write some scripts, think again. The next time you upgrade and your code is catapulted into an internationalized shell, terminal or window manager, it will break. My Perl one-liner for random e-mail signatures stopped working for this very reason on Red Hat 8.0, and in almost three months, no one on three different lists could suggest a solution.

The character encoding that is becoming the standard on Linux is Unicode UTF-8 [see "Unicode" by Reuven M. Lerner in the March 2003 issue of *LJ*]. The good news is that it can represent all characters existing on the planet, so no other encodings are necessary. Figure 4 shows Emacs, as packaged in Red Hat 8.0, dealing with all kinds of symbols and characters. The bad news is that because non-ASCII characters take more than 8 bits and sometimes much more space on screen, many deeply ingrained dogmas, starting with the equation "1 character = 1 byte", simply disappear. The right resources to deal with this, apart from the Linux Unicode HOWTO, are the mini-guide to "UTF-8 soft and hard conversion" and the UTF8_STRING mechanism that aims to preserve the X cut-and-paste system in a UTF-8 world. At a higher level, programming for internationalization, regardless of the operating system, is the goal of Openi18n.org.

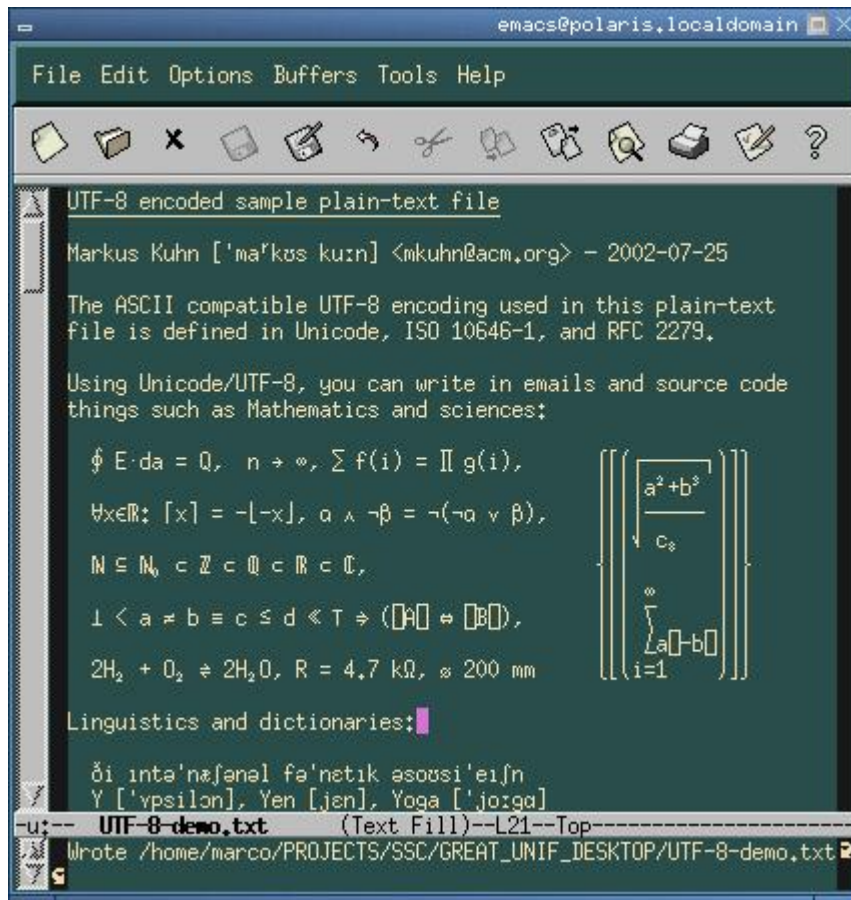


Figure 4. Emacs as Packaged in Red Hat 8.0

Menus and Icons

What is left to achieve a nice GUI? Menus and icons. A desktop entry standard that describes, regardless of the environment, how to build menus, how to launch each application and so on, already exists at freedesktop.org. It does have some limitations, namely the lack of a common place where .desktop files should be put and the hardwiring of menus, coming from the fact that they simply mirror how these files are located on disk. A virtual folder extension to the standard is being written to overcome these limits. Another specification with a similar scope is available to standardize icon locations and theme selection.

Ease of Installation

Yes, if you distribute the source, everyone can compile and install your program, but why make it hard for others to figure out why the program can't find which libraries are installed? Why hard code things so they will work only on one distribution? The Linux filesystem hierarchy from the LSB group is your friend here, whatever application you plan to write.

Conclusion and Credits

I have nothing against pure KDE or pure GNOME. I only hope that the next generation of desktop applications will make it easier for everyone to build his or her own environment from any combination of programs without sacrificing real functionality and performance. The methods and tools described here are a good way to build such applications, and I am grateful to their developers. Many thanks also to Havoc Pennington, Keith Packard, the members of the kde-devel list and everyone who answered on the linuxjournal.com web site to help me in writing this article.

Resources

email: m.fioretti@inwind.it

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

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

[Advanced search](#)

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

Fixing Photo Contrast with The GIMP

Eric Jeschke

Issue #108, April 2003

Fix the too-dark areas in your photo without washing out the light areas.

In my last article (*LJ*, February 2003), I described how to improve candid flash photos by removing red-eye using the GNU Image Manipulation Program (GIMP). In this article, I present another GIMP gem for fixing your photographs: using a digital split neutral density filter to repair bad pictures resulting from shooting high-contrast scenes.

The human eye is a remarkable image capture instrument. It is able to view a scene with a large dynamic range (range of luminosity or brightness) and to discern detail in both bright highlights and dim shadows. Dynamic range in photography is often measured in stops, where each stop represents a doubling or halving of light. Humans can discern detail in a scene with about 14 stops of dynamic range. Film and digital capture sensors are not as adept. Slide film typically can handle around 5-6 stops. Detail in areas below the lower limit is blocked up into dark shadows, and detail above the upper limit shows up as blown-out (completely white) highlights. Negative film does a bit better at 9-10 stops, and some high-end digital cameras (DSLRs) can do even slightly better than that. Typical consumer digicams fare somewhere in the lower middle of the pack and capture about 6-9 stops of detail, depending on the bit depth used in the digital capture process, the sensor size and a few other factors.

Knowledgeable photographers often have dealt with the limited dynamic range of their equipment by trying to compress the dynamic range of the scene they are photographing using fill-flash, lighting or reflectors to light up shadows or special filters, such as a split neutral density filter (sometimes also referred to as a graduated neutral density filter) to darken the highlights. An example of such a filter is shown in Figure 1. It is an accessory you can attach to the front of your lens. It has a clear side and a dark gray side, with a small continuous transition zone dividing them. The dark part of the filter has the effect of reducing light by 1 stop, 2 stops or more, depending on the strength of the

filter. When the camera is set up for a high-contrast shot (e.g., a sunset), the filter is positioned in front of the lens so that the dark part covers the highlights (e.g., the sky) and the clear part covers the rest of the image (e.g., everything below the skyline). The photographer then can meter the exposure for the shadows. If the filter is positioned correctly, the metering is accurate, and the photographer has knocked on wood, thrown a pinch of salt over his shoulder and said a short prayer, the whole image will come out properly exposed.



Figure 1. A Split Neutral Density Filter

Most casual shooters won't be bothered to carry around split neutral density filters and use them. In such situations, a compromise exposure is the only real option. A typical programmed auto-exposure metering system often will set an exposure that takes the middle road, losing detail at both ends of the luminance range. If you're willing to control the exposure yourself, follow a rule of thumb that is oft-repeated by photographers shooting slide film: expose for the important highlights. It often will be possible to rescue some detail from the shadows later, but once highlights are blown out, there's nothing that can be done to recover that detail. Remember that the rule says "important highlights". If you are taking a picture of a sunset, you want to preserve the texture and detail in the clouds, which are brilliantly lit by the setting sun; if your main subject is a moose standing in a field at sunset then you'd probably rather have the detail in the moose's fur, and let the cloud detail fare as it will.

Although you can't recover detail that is completely clipped in such exposures, it is often possible to tweak an image to rescue a fair amount that is lurking in the highlights or shadows. The process in traditional wet-film processing is called dodging and burning. When making a print from a negative, parts of the paper are exposed more or less than the rest to hold details in highlights or

pull detail from shadows. These sort of machinations used to be reserved for advanced darkroom enthusiasts. However, now anyone with a copy of The GIMP can do all of this and more with considerable ease.

Let me illustrate with the following example: a Utah sunset, shown in Figure 2 loaded into a GIMP window. I had followed the sage advice and exposed for the clouds and highlights on the cliff face and allowed the foreground to go quite dark. Using the LAB decompose plugin, I can decompose this RGB image into the LAB constituents. Of these, the L channel shows the full range of luminance values carried in the image. As you can see from Figure 3, there is a considerable amount of detail in the foreground trees, which in the original image look almost completely blocked up. This is good, but how do I pull out this detail, while retaining the beautiful detail and color of the cliffs and clouds?

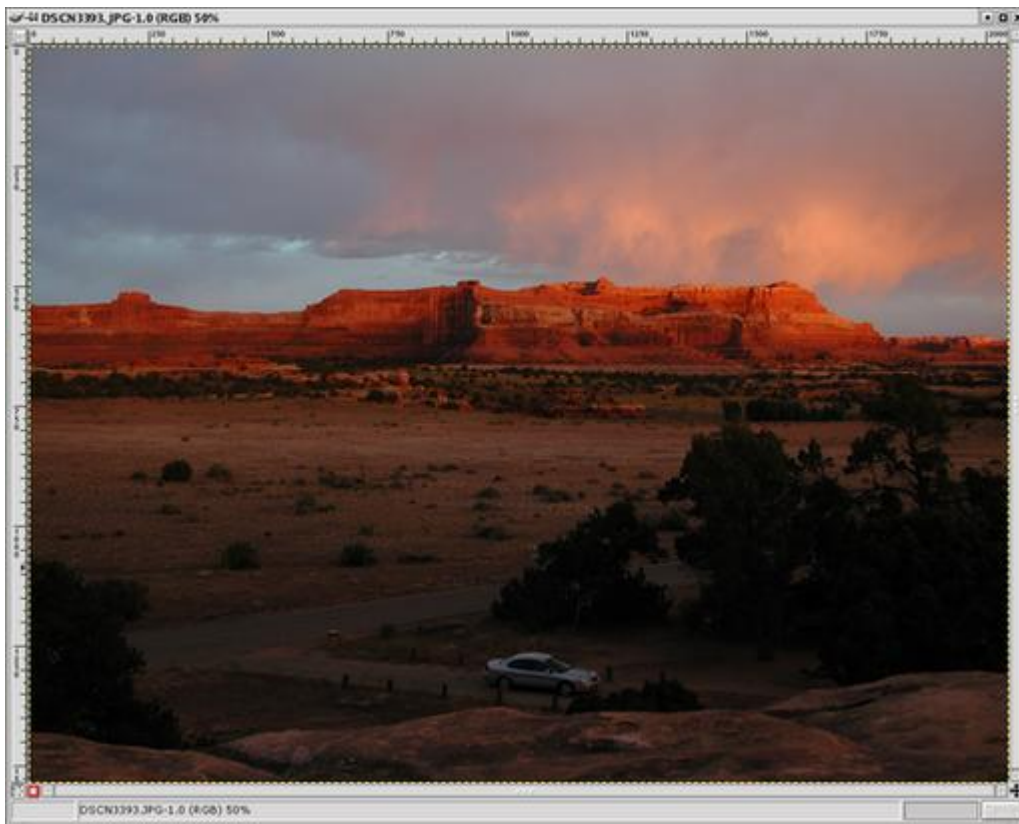


Figure 2. A Utah Sunset Photo Loaded into a GIMP Window

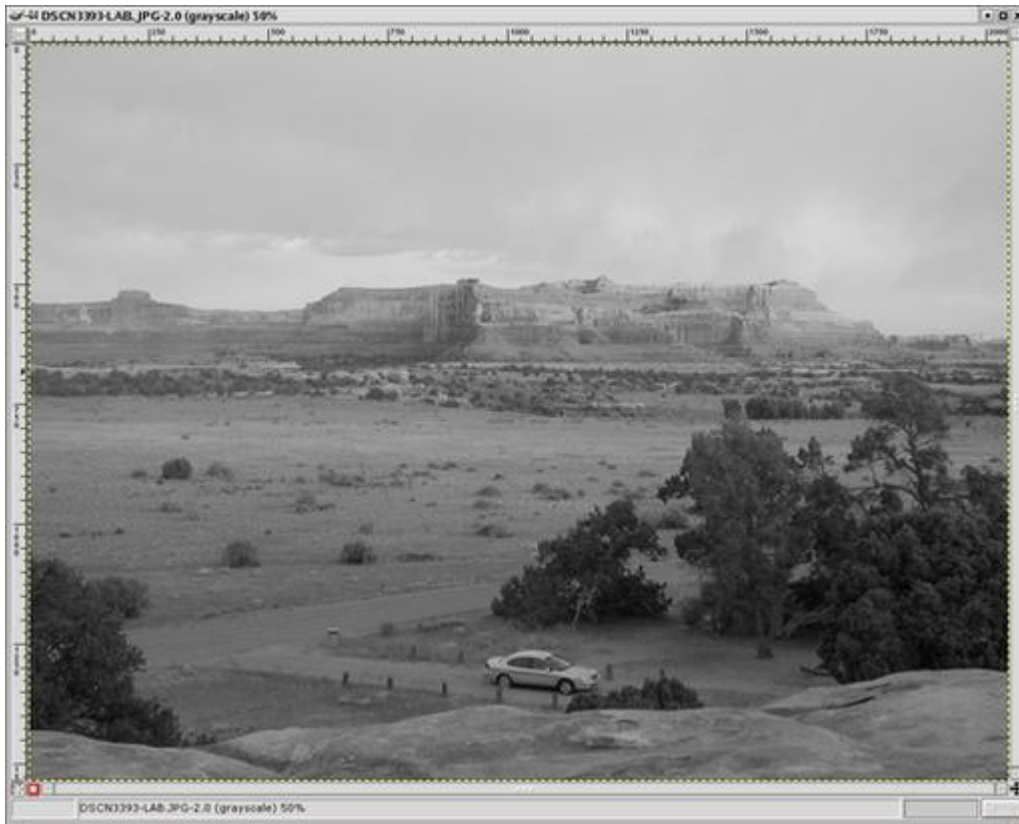


Figure 3. Using the LAB Decompose Plugin

The technique for rescuing that shadow detail is a bit like the digital equivalent of using a split ND filter. I combine two versions of the same scene, where each version has been optimized for either highlights or shadows. The technique makes use of layers and layer masks in The GIMP, so it is important to have a basic understanding of what these are beforehand. The next section introduces these concepts and provides a high-level overview of how the overall technique works.

Layers and Layer Masks

All images in The GIMP can be composed of one or more layers. When an image is first loaded, it occupies the default Background layer, as shown in Figure 4. You can add additional layers on top of the background layer. These upper layers can contain anything you want. Frequently, you'll want to create upper layers that are different versions of the same image. This is most easily accomplished by duplicating a layer, such as in Figure 5 where I duplicated the Background layer. Each layer can be manipulated independently of the others. In Figure 6, I used a Levels adjustment on the upper layer to lighten it.



Figure 4. The Default Background Layer



Figure 5. Duplicating a Layer



Figure 6. Using a Levels Adjustment to Lighten a Layer

Layers can be combined in various ways to produce a single image, as if you were looking through the top layer down to the bottom. One way of doing this is to reduce the opacity of part or all of the upper layers. The opacity of a layer can be changed from 100% (opaque) to 0% (completely transparent) or anywhere in between. It also is possible to make parts of the same layer have different opacities (or levels of transparency, if you prefer to think of it that way). Again, there are many ways to do this, but one of the most flexible ways is with a layer mask. A layer mask can be added to a layer and becomes one of its attributes. It is a gray-scale image that is the same size as the layer. The layer mask has the effect of varying the opacity of each pixel in the layer according to each corresponding pixel value in the mask. A black pixel in the mask makes the corresponding pixel in the layer completely transparent. A white pixel makes it completely opaque, and any value in between calculates a percentage of opacity between these two extremes.

I'm sure you are beginning to see the possibilities. In Figure 7 we see a useful layer mask created for the upper layer in this image. Once the layer mask is added, it is filled with a gradient, which has the effect of carefully blending the upper layer image from completely transparent to totally opaque, as shown in Figure 8. Finally, the image can be flattened for output, as shown in Figure 9.

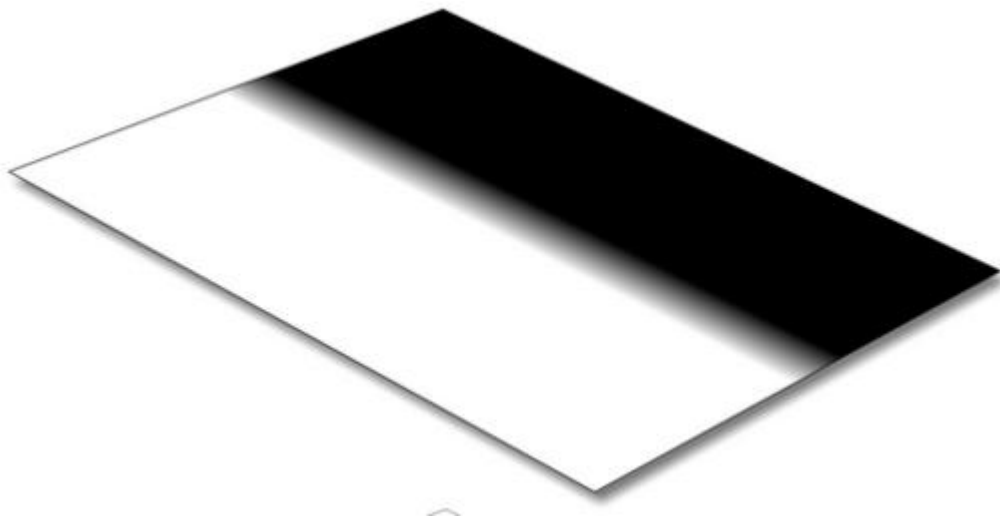


Figure 7. A Layer Mask



Figure 8. Using a Gradient



Figure 9. Flattening the Image for Output

Layer masks are only one of the many attributes a layer can have. Other attributes, such as the blend mode, also affect how a layer combines with the other layers below it. Further research into how layers work will pay off with big rewards in your GIMP image editing abilities.

The Nitty-Gritty

Now that we have layers and masks under our belt, let's get to the details of the technique. Most of the menus in The GIMP are accessed by clicking the right-most mouse button in an image window. In the description that follows, a right-click is abbreviated RC. If I describe a GIMP action to invoke I will mention the series of menus or a keyboard shortcut in parentheses. For example, Open the image (RC-->File-->Open), means right-click in the image window and choose File, then from that menu choose Open. If a keyboard shortcut makes more

sense I'll list the combination of keys to press; e.g., copy the image (Ctrl-C) means press and hold the Control key and press C.

In the original image, open the Layers dialog (Ctrl-L). In that dialog, right-click on the Background layer name and select Duplicate, or use the button circled in Figure 10. Now double-click on the duplicate layer and rename the new layer ND Filter. This step is not strictly necessary, but it is helpful to prevent confusion about what is on each layer, especially if you open this file again six months later and are trying to figure out what you did. At this point, your Layers dialog should look something like Figure 10.

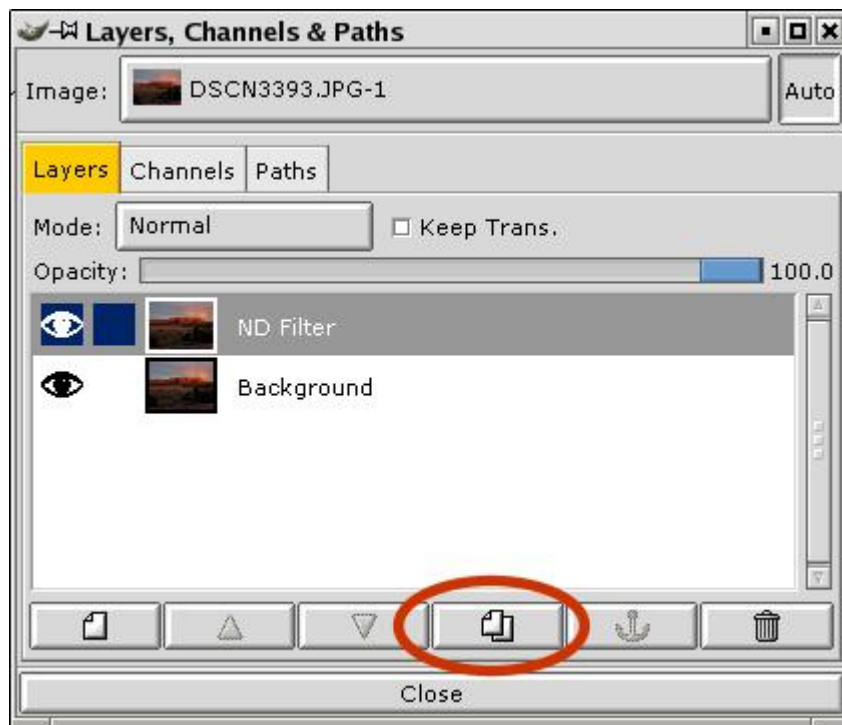


Figure 10. Layers Dialog

In the Layers dialog, select the ND Filter layer. Go to the image window and perform any editing you want to enhance highlight or shadow detail. The Levels (RC-->Image->Colors-->Levels) or Curves (RC-->Image-->Colors-->Curves) operations are good choices for this, but you can also use Brightness/Contrast (RC-->Image-->Colors-->Brightness and Contrast), the dodge/burn tools or anything else that works for you. We are working on the duplicate layer so don't worry about the good part of the image; let it go too dark or too light. Concentrate on the area that needs improvement. In this case I've used Levels (moved the middle slider to the left a bit, as shown in Figure 11) to lighten the whole image until the foreground is about where I want it.

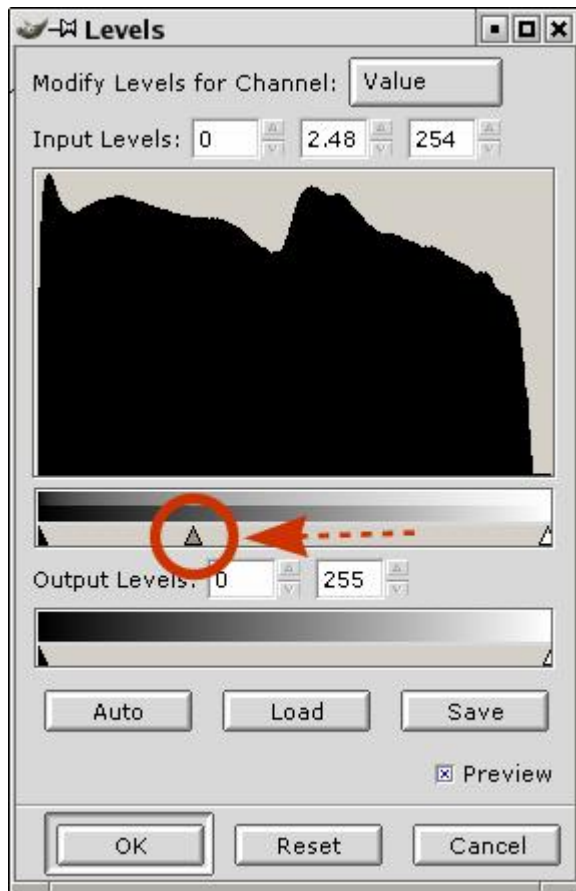


Figure 11. Using Levels to Lighten an Image

At this point you will be in the position illustrated by Figure 6, where you can see your lightened image, which obscures the original image below it. We now want to add a layer mask that will reveal the upper part of the lower image. In the Layers dialog, right-click on the ND Filter layer and select Add Layer Mask. In the Add Mask Options dialog, select White (Full Opacity) and click OK. Finally, click the eye next to the Background layer in the Layers dialog to turn off visibility of the Background layer. Your Layers dialog should now look like Figure 12, with a little white layer mask icon next to the layer image icon on the ND Filter layer.

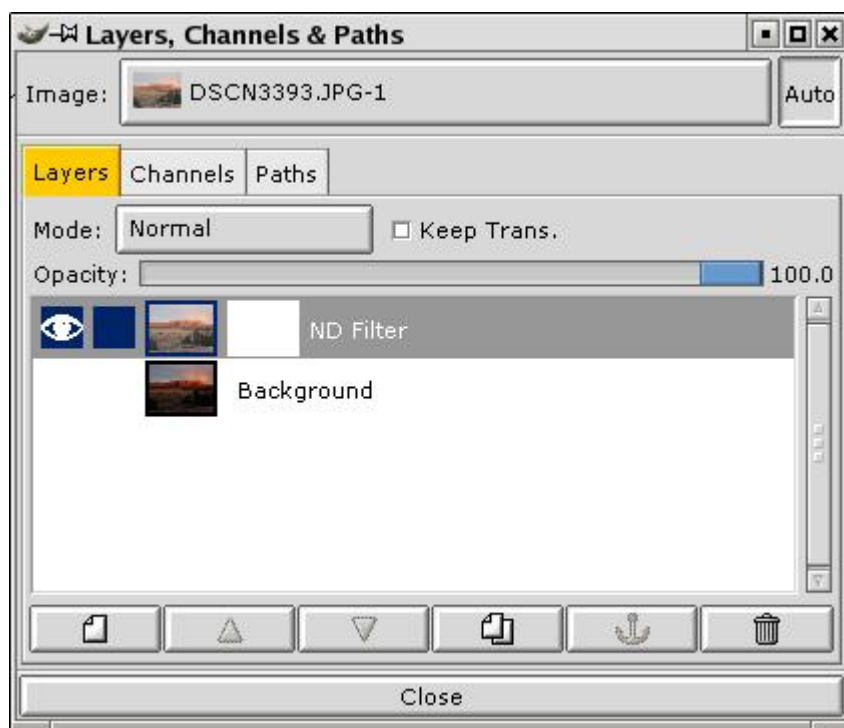


Figure 12. Turning off the Visibility of the Background Layer

Go up to the main GIMP toolbox window and select the gradient fill tool, circled in Figure 13. Go back to the image window and click-drag a line in the angle and direction that you want the split between the upper and lower layers (lower layer at the beginning of the drag, upper layer at the end). The length of the line that you drag determines how graduated or abrupt the transition will be, and the resulting blend. Experiment to get a feel for it, but generally you will want a short stroke centered over the transition zone (e.g., horizon). Because we turned off visibility of the background, your feedback will be immediate, as shown in Figure 14; a portion of the image should disappear just past the transition zone. If you didn't get the split at the right place simply click-drag another line, and the new gradient fill will replace the old. To get an idea of what you are doing, refer back to the gradient in Figure 7. You can see a miniature version of your gradient in the layer mask icon in the Layers dialog. Remember, the gradient fill affects the transparency of the upper layer. White is opaque; black is transparent, and anything in between is some degree of translucent.



Figure 13. The Gradient Fill Tool

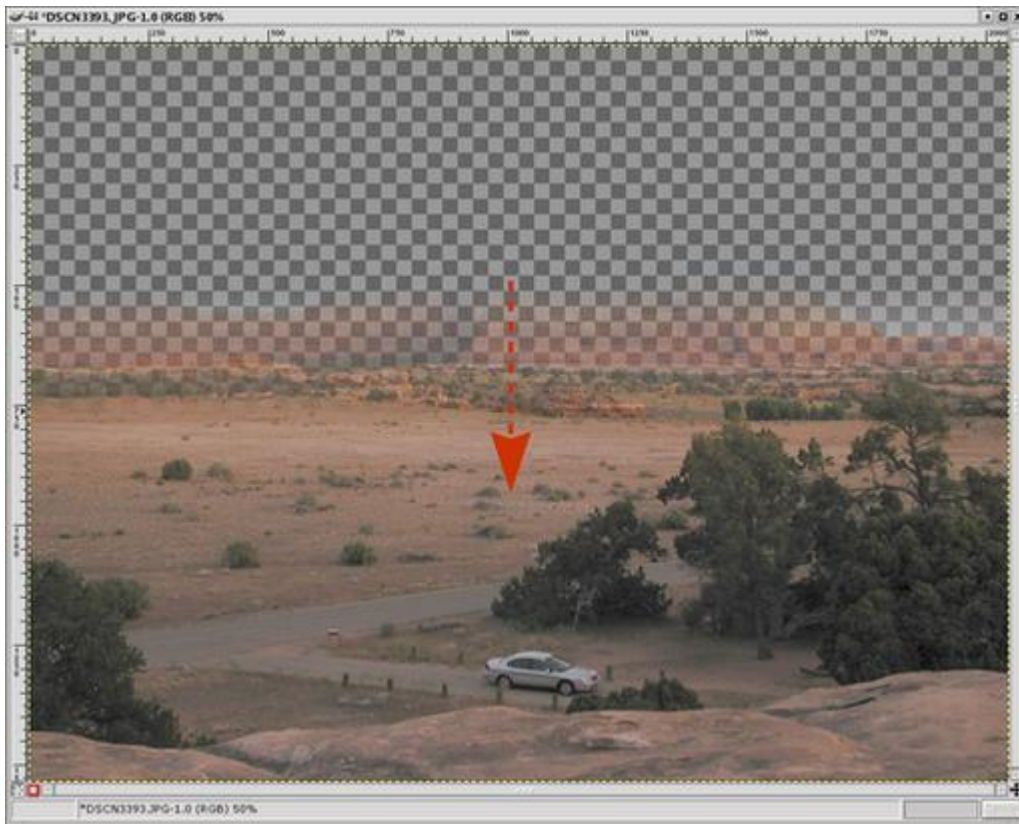


Figure 14. Applying the Gradient Fill Tool

Time to check your handiwork: click the eye next to the Background layer in the Layers dialog to turn visibility of that layer back on. Behold—your combined image with the best exposure of both worlds. Now, click the eye next to the ND Filter layer in the Layers dialog off and on to view the effect quickly with and without this digital split ND Filter. Nice, eh? Figure 15 shows the final result.

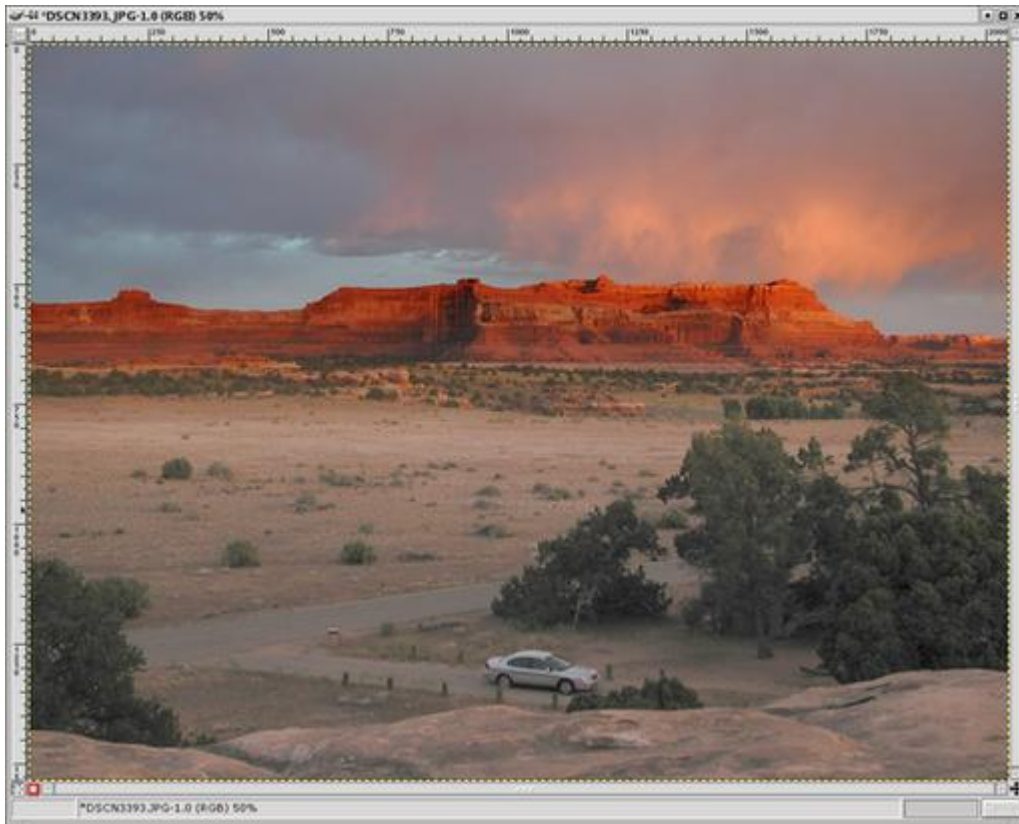


Figure 15. The Final Result

The real beauty of this approach is that our original image is untouched on the Background layer. The lightened image on the second layer did not require any painstaking selection to affect only the required areas, and the layer mask (in this case) was trivial to create. For maximum flexibility, you can save this image in The GIMP's native XCF format, and it will retain all of the layered structure. This allows you to go back in and make additional adjustments easily, safe in the knowledge that your original unmodified image is conveniently available on the Background layer if you ever need to redo the upper layer or the layer mask. When you are ready to export the image to one of the more common image formats, such as TIFF or JPEG, it will flatten the image as shown in Figure 9.

Knowing how the layer mask works, you can visualize how to apply the technique to images that have much more complicated transitions between the highlights and shadows (e.g., a dark mountain sticking up into the sky). The layer mask can be manipulated like any other gray-scale image to force the upper and lower layers to blend together wherever you like and in whatever manner you like.

I hope you find this technique useful. I'm guessing that it will rescue many vacation sunset photos—perhaps one from the Geek Cruise?



email: jeschke@mano.uhh.hawaii.edu

Eric Jeschke (eric@redskiesatnight.com) holds a PhD in Computer Science from Indiana University and has worked as a software engineer, university professor and freelance consultant. He lives in Hawaii with his wife, kids and an overweight cat. Eric enjoys his family, outdoor adventures, taking photographs and running Linux.

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

[Advanced search](#)

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

[Advanced search](#)

Programming under GNUstep—An Introduction

Ludovic Marcotte

Issue #108, April 2003

With GNUstep, you can develop a new application or port one from Mac OS X.

This article provides an introduction to GNUstep development. It guides the reader through installing GNUstep and developing a small application. The Objective-C language, a true object-oriented superset of ANSI C, is used in this article. Its syntax is simple, unambiguous and easy to learn.

A Short History

OpenStep, which was proposed as an open standard by NeXT Computer, Inc., in 1994, is a collection of advanced object-oriented APIs designed for rapidly developing applications in the Objective-C language. It was designed to be implemented independently of the computer's operating system. To this end, there were implementations of the standard for Mach, Microsoft Windows (NT and 95), Sun Solaris and HP-UX.

In the early 1990s, GNUstep, a free implementation of the OpenStep standard, was born. GNUstep aims to be a fully compliant OpenStep implementation, supporting a wide range of operating systems, while being entirely free. GNUstep is released under the terms of the LGPL license. GNUstep currently works well on Linux, Solaris and most BSDs, and there is also preliminary support for Microsoft Windows.

As was widely reported at that time, Apple bought NeXT in 1996. Now, with Mac OS X, Apple is moving forward with Cocoa, an extension of the OpenStep API. While creating Cocoa, Apple added new classes that enrich the API. GNUstep is partially supporting those new features, and better support is being added every day.

GNUstep also offers a well-defined separation between the user-interface classes (part of the application kit) and the underlying windowing system. This

layer, called gnustep-back, provides implementations for various windowing systems, allowing gnustep-gui to work properly under various back ends. Back ends currently are being developed for the X Window System, Microsoft Windows and Ghostscript. Figure 1 shows the different layers of GNUstep.

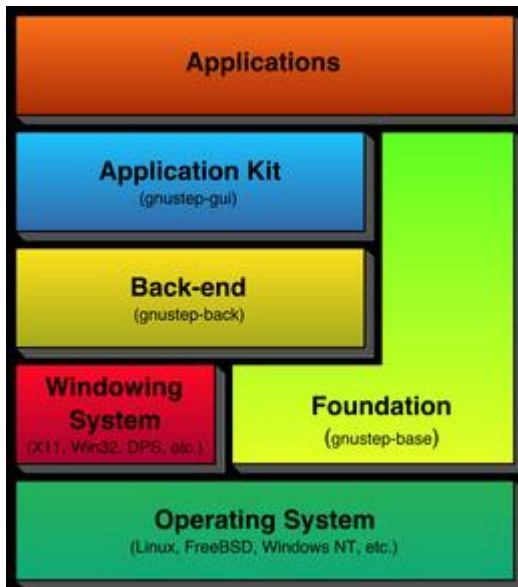


Figure 1. The Different Layers of GNUstep

GNUstep also offers bindings to other languages, like Java (using JIGS) and Ruby (using RIGS), allowing developers to create applications using those languages. Additionally, GNUstep offers powerful scripting capabilities through StepTalk, which allows you to create scriptable applications or servers using Smalltalk. Finally, GNUstep offers an extension so that Guile, a version of Scheme, can be used as a scripting language.

Installing GNUstep

In order to compile GNUstep under Linux, that the following programs and libraries (with headers) need to be installed (versions indicated or higher): GNU make 3.75, GNU libc 2.1.2, GNU gcc 3.0.3, XFree86 4.1, ffcall 1.8, OpenSSL 0.9.6c (not required but recommended), libtiff 3.5.5 and libxml 2.2.3.

Once all those requirements are satisfied, you are now ready to download and compile GNUstep. We are going to install an unstable release of GNUstep because new features have been added recently that are worth using and mentioning, especially regarding Gorm—the Graphical Object Relationship Modeler. Gorm is a clone of OpenStep's excellent interface-builder application. Gorm allows the developer to create user interfaces quickly from a palette of standard objects.

Get the following releases from the GNUstep web site: GNUstep make 1.5.1, GNUstep base 1.5.1, GNUstep gui 0.8.3, GNUstep back 0.8.3 and Gorm 0.2.0.

Once you have downloaded those releases, you are ready to install GNUstep. You must first install GNUstep make, base, gui and finally, back. Look at the INSTALL file of each release carefully for installation instructions. For detailed instructions on installing GNUstep, refer to the GNUstep Build Guide for Unix Systems (see Resources).

Once GNUstep is compiled and installed, load the proper set of environment variables by executing a shell script (adjust the path, if necessary).

Bash users would type:

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

And, C-shell users would type:

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

To compile and install Gorm, simply uncompress the file and type (as root):

```
# cd Gorm  
# make  
# make install
```

Developing a Small Application

Now that GNUstep and Gorm are compiled and installed, you can develop your small application, a simple TIFF image viewer. This application will use the Model-View-Controller (MVC) design. The MVC design pattern separates a software component into three pieces: a model, a view and a controller. Figure 2 shows how the MVC pattern is used in our small application.

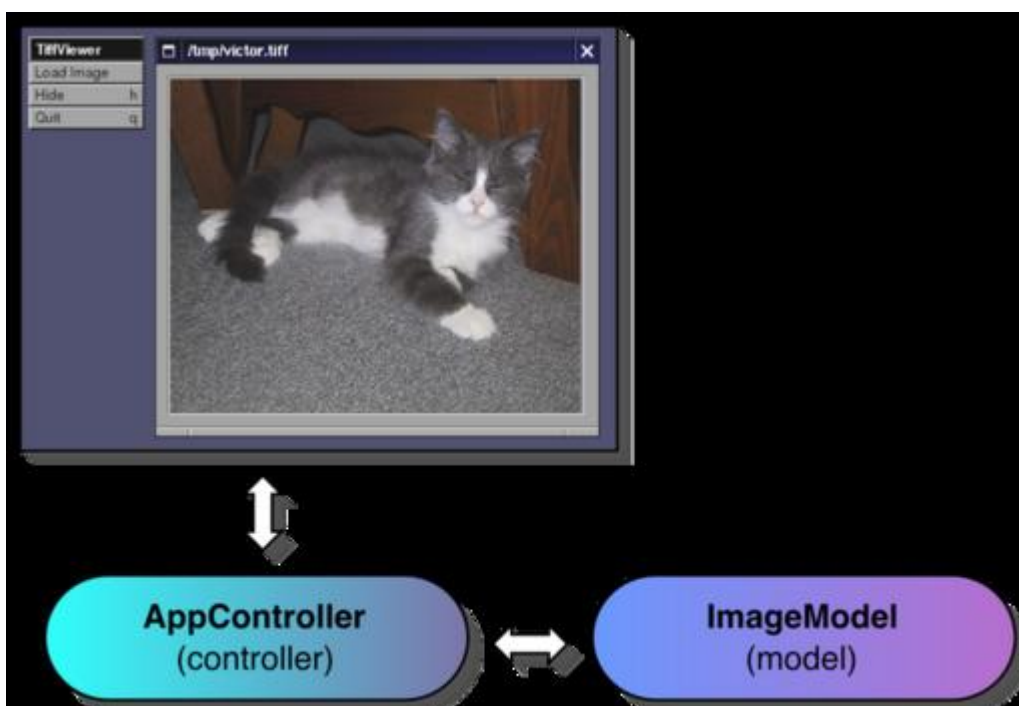


Figure 2. Using the MVC Pattern

Under GNUstep, use Gorm to create the views of an application. This application allows you to create the user interface of your application. Once created, the user interface will be saved as an archive file (containing a serialized object graph) in its own subdirectory. To start Gorm, simply type:

```
# openapp Gorm.app
```

Once Gorm has started, show the Inspector and the Palettes windows by clicking on the Inspector... and Palettes... menu items from the Tools menu. Now, create a new application by clicking on the New Application menu item from the Document menu. Once those steps are completed, your desktop should look like Figure 3.

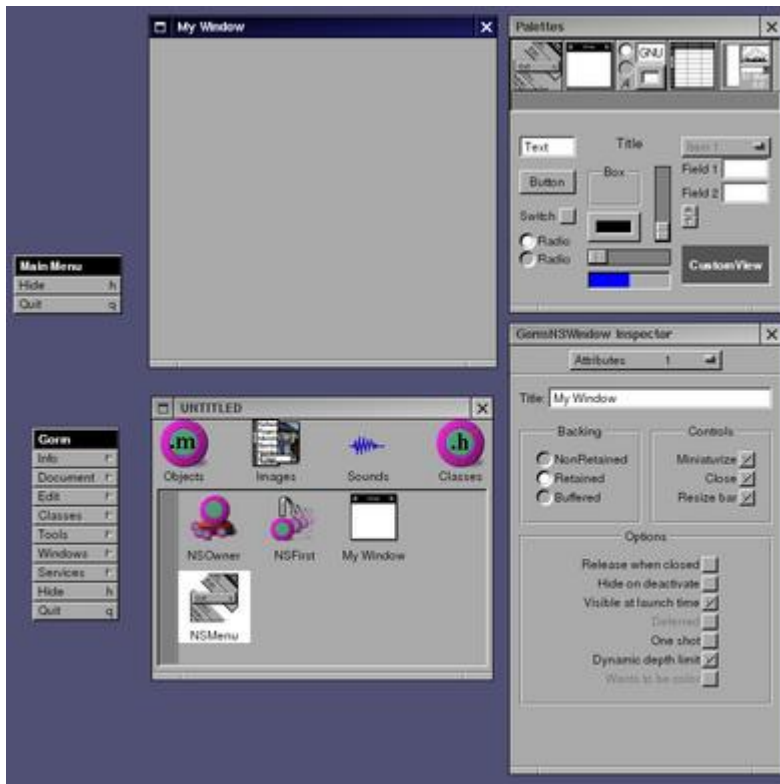


Figure 3. Creating a New Application

Let's examine Figure 3 in order to explain the different elements. Table 1 shows the different elements, and their roles.

Table 1. Parts of the Gorm Interface

From the Palettes window, drag an NSImageView object to the empty window (My Window) and drop it. Figure 4 shows the icon that represents the NSImageView object.



Figure 4. NSImageView Object

Now, click on the NSImageView in the window, and the inspector window will update itself so you can set the correct attributes to this object. Set the border to Line Border and the scaling to To Fit. The Inspector window for the attributes of the NSImageView should look like Figure 5.

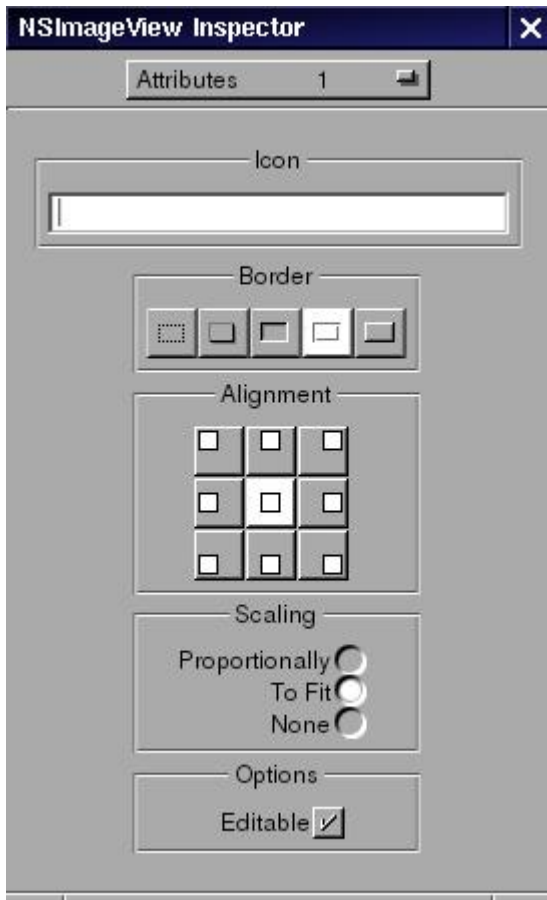


Figure 5. The Inspector Window for the Attributes of NSImageView

Now, from the Inspector window, select the Size item from the combo box at the top to set the auto-sizing properties of the NSImageView object. Setting the auto-sizing properties to NSImageView will allow the object to resize properly if the user resizes the window. Set the parameters by clicking on the lines inside the Auto-sizing box to look like those in Figure 6.

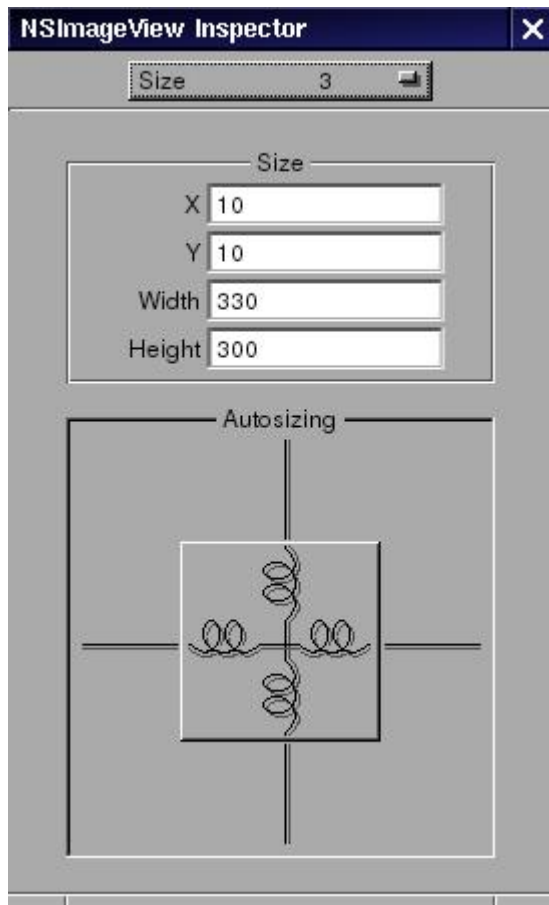


Figure 6. Setting the Auto-Sizing Parameters

Next set the window attributes. Click on My Window in the File window. Set the title to Tiff Viewer in the Inspector window. The Inspector window for the attributes should look like Figure 7.

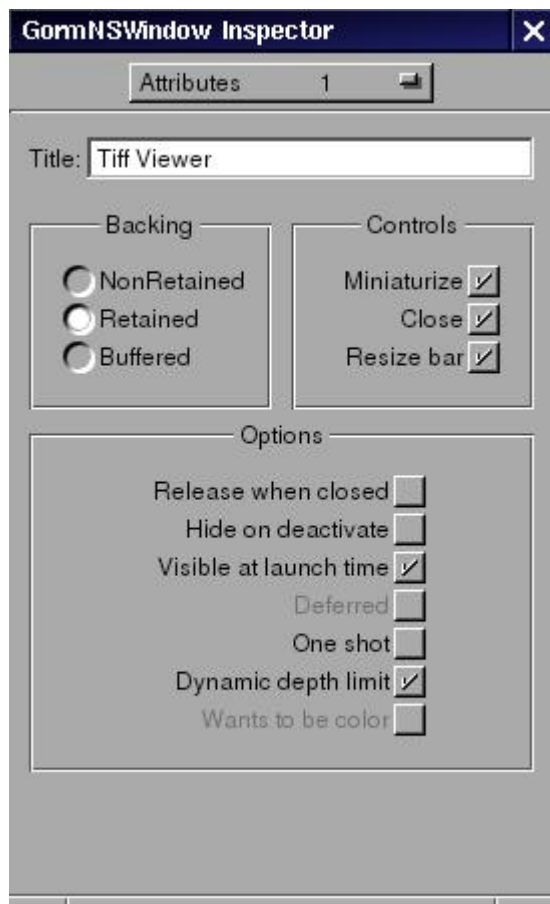


Figure 7. Setting the Window Attributes

You need to add a menu item so you can open an image file. To create a new menu item, choose the menu palette from the Palettes window, drag the Item menu to the Main Menu (above the Hide menu item) and drop it. Click on the newly created Item menu option in the Main Menu, set its title from the Inspector window to Load Image and press Return.

Now, you need to create a controller class that will interact with the user interface. Click on the Classes icon from the File window and select the NSObject item from the tree view. Click on the Classes menu item in the Gorm menu, and select Create Subclass.... In the Classes tree view in the File window, you will now find a NewClass entry. Double-click on the NewClass item and replace the text with "AppController".

Next, you need to create an outlet for your UIImageView object. An outlet is an instance variable that refers to an object—in this particular case, the UIImageView object. To add an outlet to your AppController class, click on the Outlet icon to the right of the AppController class (in the File window); click on the Classes menu item in the Gorm menu, and select Add Outlet/Action.... In the Classes tree view in the File window, you will now find a newOutlet entry as a child item of AppController. Double-click on the newOutlet item, and replace the text with "imageView". Repeat these steps to add a window outlet.

After creating the outlet, you need to add an action to your controller. An action is a target-action method. To add one to the controller, click on the Action icon to the right of the ApplicationController class (in the File window); click on the Classes menu item in the Gorm menu, and select Add Outlet/Action.... In the Classes tree view in the File window, you will now find a newAction: entry as a child item of ApplicationController. Double-click on the newAction: item, and replace the text with "loadImage:".

You are now ready to instantiate your ApplicationController class in order to produce a particular object from its class template. You need to instantiate your controller class because Gorm connects live objects. Select the ApplicationController item in the File window's tree view, and click on the Instantiate menu item from the Classes menu.

Now you can now connect your live objects (in this case, NSImageView) to your outlets and set the action. To connect the NSImageView object to your ApplicationController instance, click the Objects icon in the File window, then, while pressing the Ctrl key, click and drag from the ApplicationController icon (from the File window) to the NSImageView object in the Tiff Viewer window. Click on the imageView item in the Inspector window, and click on the button Connect. This connects the NSImageView live object to the imageView outlet. To connect the window outlet, while pressing the Ctrl key, click and drag the ApplicationController icon (from the File window) to the My Window icon (from the File window). Now, click on the window item from the Inspector window, and finally, click on the Connect button. Figure 8 illustrates the connection being made between the window object and its outlet.

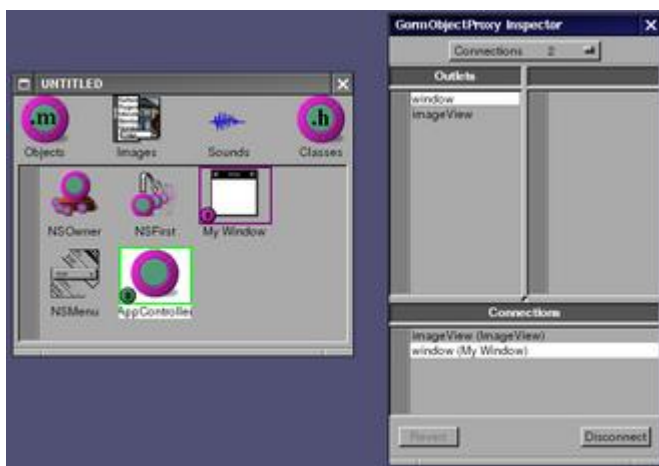


Figure 8. The Connection Being Made between the Window Object and Its Outlet

To set the action of the Load Image menu item, Control-click and drag on the menu item to the ApplicationController instance icon in the File window. Then, click on target in the Inspector window, select the loadImage: item, and click on the Connect button. This sets the action of the Load Image menu item to

loadImage:. So, when the Load Image menu item receives a click from the user, the loadImage: method will be invoked.

Now, save everything by choosing the Save menu item from the Document menu. Give it the name MainMenu. This creates a MainMenu.gorm directory, holding the archived view of the application.

Finally, create the ApplicationController class' source files by selecting the ApplicationController class entry in the Classes tree view and choose the Create Class's Files... from the Classes menu. Leave the names as they are and simply click on the OK button. This creates two files: ApplicationController.m and ApplicationController.h.

Now, quit Gorm, open your favorite editor and modify ApplicationController.h so it looks like Listing 1. The complete source code with comments is available; see Resources.

Listing 1. ApplicationController.h

Then, modify ApplicationController.m (the implementation of the ApplicationController class) so that it looks like Listing 2.

Listing 2. ApplicationController.m

Now, create a very basic Model for the application—ImageModel. This small class will hold the current displayed image. Now create the implementation of the ImageModel class as shown in Listings 3 and 4.

Listing 3. ImageModel.h

Listing 4. ImageModel.m

Now, create a small property list, shown in Listing 5, in which you specify the application name, description and the name of the main Gorm file to load upon startup.

Listing 5. TiffViewerInfo.plist

Then, create a small GNUmakefile in order to compile and link your small application. The GNUmakefile should look like Listing 6.

Listing 6. GNUmakefile

Finally, compile and start the small application:

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

Once the application starts, click on the Load Image menu item and select a TIFF file. It should display properly in the window, as shown in Figure 9.

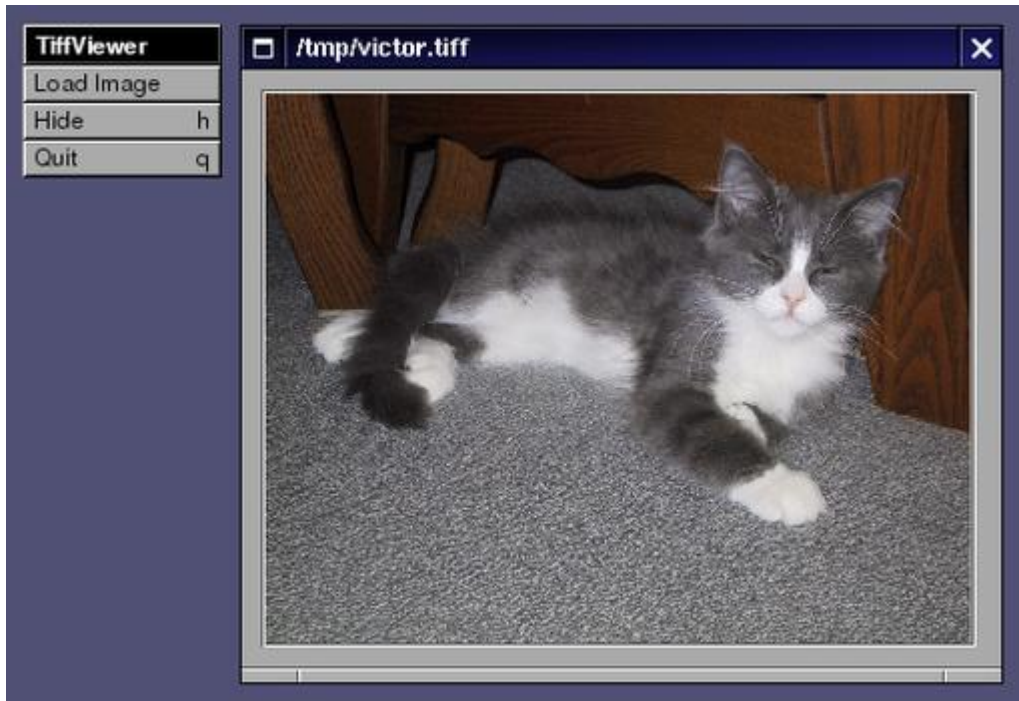


Figure 9. Displaying an Image

Porting to and from Apple Mac OS X

When porting an application from GNUstep to Mac OS X (or the other way around), some important things must be considered. For example, when porting to Mac OS X, you have to redo the user interface using Interface Builder under Mac OS X. The following steps are needed to port the application to Mac OS X:

1. From the File menu of Project Builder, select New Project... and select Cocoa Application. Click on the Next button.
2. Specify the project name (Tiff Viewer) and the project location. Click on the Finish button.
3. Now, select the Classes node and add the files ApplicationController.m and ImageModel.m from the Project menu.
4. Expand the Other Sources node and delete the main.m file.
5. Expand the Resources node and double-click on the MainMenu.nib node. This will start Interface Builder.
6. Much like you did under Gorm, drag and drop the UIImageView in the empty window and the Load Image menu item in the File menu.

7. Because the controller class (AppController) was created in the previous section, you simply can reuse it under Mac OS X. In Interface Builder, from the Classes menu, choose Read Files.. and select AppController.h.
8. From the File window, click on the Classes tab, select AppController and from the Classes menu, choose Instantiate AppController.
9. Now, connect the outlets and the action like we did under Gorm.
10. Save the modified interface, and quit Interface Builder.
11. From the Build menu in Project Builder, choose Build and run. This will compile and launch the application.

Once the application is launched, choose Load Image from the file menu and select a picture to show. The final result should look like Figure 10.



Figure 10. The Same Application Running under Mac OS X

As you can see, we have ported the application without rewriting a single line of code. Even if the application is quite simple, complex applications can be developed under GNUstep and easily ported to Mac OS X. Affiche and GNUMail.app are good examples of applications that are portable between GNUstep and Mac OS X.

Going the other way, more care should be taken when porting applications from Mac OS X to GNUstep. First, you have to redo the user interface of the application using Gorm. Secondly, GNUstep currently does not provide an implementation of some Cocoa classes like NSToolbar, NSDrawer or any core foundation services. To avoid problems when porting a Mac OS X application using those unimplemented functionalities to GNUstep, you will need to use

conditional compilation. Finally, one or more GNU Makefiles must be created in order to compile the application under GNUstep.

Conclusion

As we have seen in this article, developing a GNUstep application is relatively easy. GNUstep offers a rich, clean and consistent API for developing true cross-platform applications in the Objective-C language.

New application kit back ends are being developed for Microsoft Windows, DirectFB and Ghostscript, thus allowing support for a wider range of computing environments. Also, OpenGL support has recently been added through the implementation of the NSOpenGLView class.

Finally, GNUstep-based distributions are emerging. For example, the LinuxSTEP Project aims to create a fully integrated, desktop Linux operating environment that is not bound by some of the more traditional approaches of common Linux distributions.

All listings referred in this article are available by anonymous download at <ftp://linuxjournal.com/pub/lj/listings/issue108/6418.tgz>.

Resources

email: ludovic@sophos.ca

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

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

[Advanced search](#)

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

[Advanced search](#)

The GNOME 2 Desktop Environment

Russell Dyer

Issue #108, April 2003

A look at the improvements and functionality of the GNOME 2 desktop.

The GNOME Foundation (gnome.org) released GNOME version 2.0 last summer and version 2.2 in January 2003. These releases mark a move toward a standardized desktop and a commitment to scheduled releases. GNOME has become an excellent choice for first-time and nontechnical users. "With the exception of some specialized applications, one can be fully productive in business with the GNOME desktop. That's something that has only occurred with free software in the last 12 to 18 months", says Tim Ney, GNOME Foundation executive director. With the improvements made in GNOME 2, Linux's chances of increasing its market share rose significantly.

Changes to GNOME

The transition for GNOME 1.x users to GNOME 2 will involve a few minor annoyances that can be expected when upgrading a desktop. Going from one learning curve to another is common in Linux; however, in upgrading my personal computer to GNOME 2, I was surprised to realize GNOME 1.4 had given me a certain level of comfort I did not want disturbed.

The people at the GNOME Foundation told me they were looking to streamline the desktop and cut down on the multitude of choices—to make workstations simpler and reduce the learning curves for new users. As a result, many minor applications, especially redundant ones, have either been eliminated or moved to a submenu labeled Extras. GNOME 2 also provides a more consistent look and feel from one program to another, thanks to an improvement in themes and fonts. "We're trying to strike more of a balance—setting standards while at the same time keeping the flexibility of Linux", says Havoc Pennington, GNOME developer and Foundation board member.

Despite how content some of us may be with Linux in the rough, nontechnical users don't always appreciate the struggle. To compete for larger markets, GNOME has simplified the desktop. Pennington says, "It's not about removing choices but giving you a default choice to get you started quickly and easily." So, instead of offering the user five different browsers and three different word processors, there's one of each. If you don't like the ones picked out for you—which were designed with the new GNOME 2 libraries—you can always find the RPM on the Web and load it.

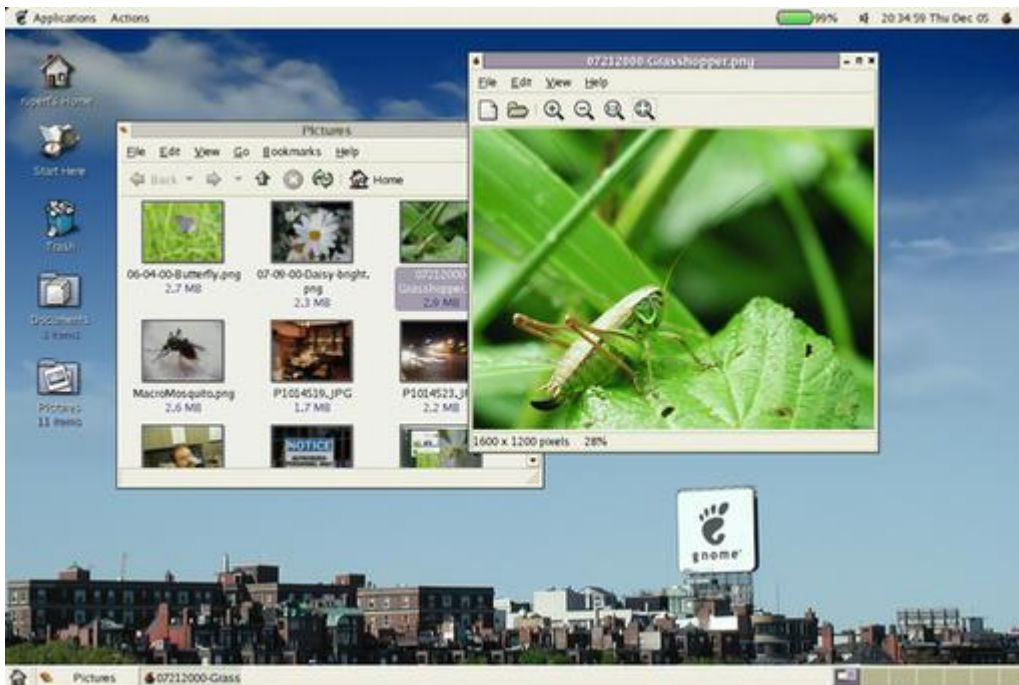


Figure 1. GNOME Desktop

Thanks to a usability study conducted by Seth Nickell of the GNOME Project, the program menus have been nicely reorganized. GNOME no longer buries utilities under several layers of submenus or within other programs. It also does not require the user to make command-line changes or to edit configuration scripts directly. Instead, GNOME 2 provides graphical interfaces for just about all system settings in easy-to-find places under main menus.

GNOME comes with many applications and utilities. Because I cannot cover all of them, I review a few key components from each section to help those new to GNOME get started. Some components are new to GNOME 2, and others have been included in GNOME for some time.

Panels

As is common with some desktop environments, GNOME provides users with panels for launching and managing programs, as well as for monitoring their systems. Panels can be placed at the top, bottom, left and right margins of the desktop. More than one panel per margin can be set up, and panels can be

floating so that they can be placed anywhere on the desktop. They can be configured to remain open, to hide automatically, or they can resize themselves as needed depending on the number of running applications. A panel also can be set up with a button for extending and retracting the panel.

Initially, one panel is set up along the top margin with a menu for launching programs. Another panel is placed along the bottom margin with icons for launching each major component of OpenOffice, which is now the default office suite for GNOME 2. It also has links to the web browser, Mozilla, and the default e-mail client, Ximian's Evolution. All things considered, these are good choices for the average Linux user and especially for nontechnical users. If you don't like these choices, however, you easily can remove or add program launchers to the panels. Simply right-click on an icon and choose Remove from the panel to remove it. To add an item, right-click on an open section of the panel, select the Add to panel menu, and pick the application you want to add from one of the submenus.

Workspaces

The bottom panel initially contains a workspace switcher or pager. This isn't unique to GNOME, but it's a useful component of the desktop, allowing you to have multiple virtual desktops running simultaneously. You can open a couple of related applications on one page and have another page with another set of applications, so you can switch between them quickly without having to minimize and maximize. For example, when using a program like The GIMP, which opens each image and tool in a separate window, it's handy to be able to switch to another workspace to check for e-mail without disturbing The GIMP windows.

Applets

Depending on your computer, certain applets already are placed on the panels, but several others may be added. By default, the clock applet is installed. However, the choices for faces are now limited to a single digital one. The old GNOME provided a half-dozen or so, including some stylish analog clock faces. For laptops, a battery monitor applet is placed on the panel. You can add a dictionary form, a weather report applet, a scrolling marquee stock ticker, modem indicator lights, an e-mail box notifier, a CD player and audio volume controls, and a floppy disk mounter—something useful for users who don't know the mount command to save their data.

Continuity

One very useful feature of a fully integrated desktop environment is the ability to copy and paste text between applications. In recent years, this has been

possible in GNOME, but it was fairly spotty and not very dependable. However, this no longer seems to be much of a problem. You now can copy from a web browser to a terminal window running vi to a word processor and so on. Resolving this bug has greatly added continuity to the GNOME environment.

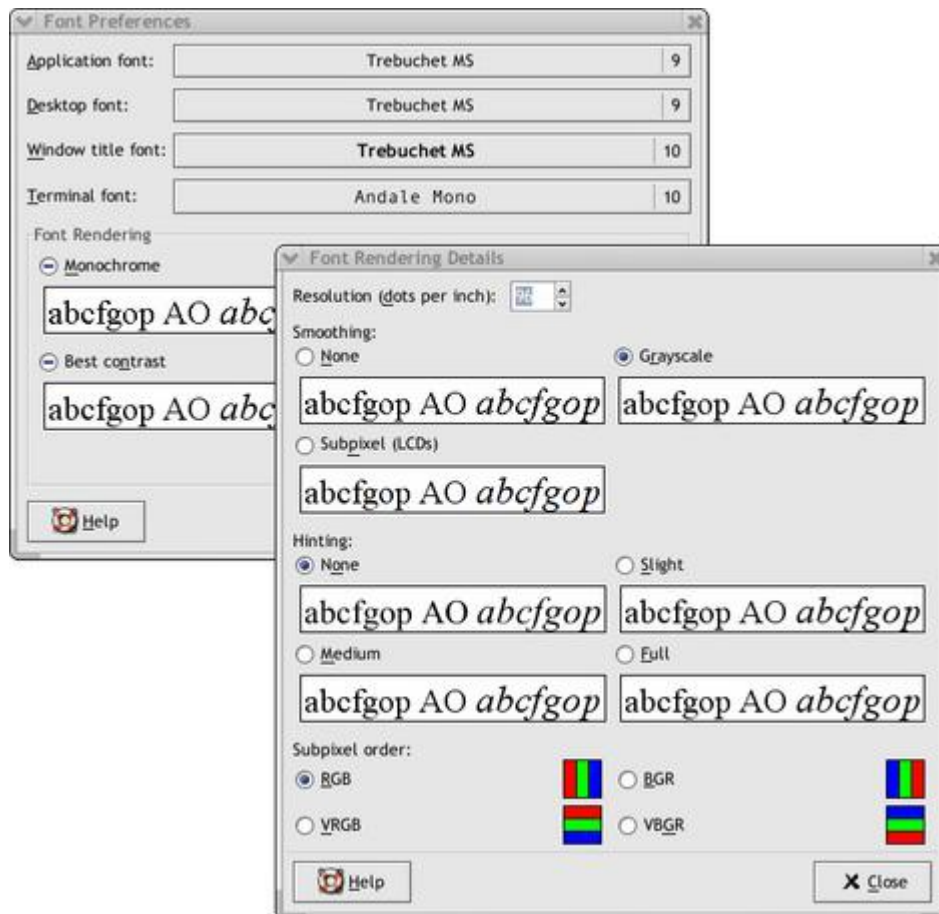


Figure 2. GNOME 2.2 Really Cleans up the Fonts

Another continuity achievement is the improvement in screen fonts. In the past, screen fonts had a jagged look. That's all changed, says GNOME Release Coordinator Jeff Waugh of Sydney, Australia:

With GTK+ 2.0 [used in GNOME 2.0], we gained Pango, a font layout, rendering and i18n library. But with GTK+ 2.2 [used in GNOME 2.2], it now supports new font configuration software written by Keith Packard, fontconfig. This really cleans up the fonts. They span all desktop applications in GNOME because everything is based on GTK+.

The result is a desktop with a consistent, professional look throughout.

Nautilus

An integral part of GNOME is Nautilus, a graphical interface for managing files and configuring Linux. The simplest way to access Nautilus is by double-clicking

on the Home icon on the desktop. Nautilus is a comprehensive drag-and-drop file manager. You can copy and move files by key strokes, by dragging and dropping a file's icon from one window to another or by right-clicking on a file's icon and selecting a choice from the pop-up menu. The pop-up menu also provides a screen for modifying permissions and ownerships. You also can now add graphical markers to icons associated with specific files to earmark them (e.g., important or personal).

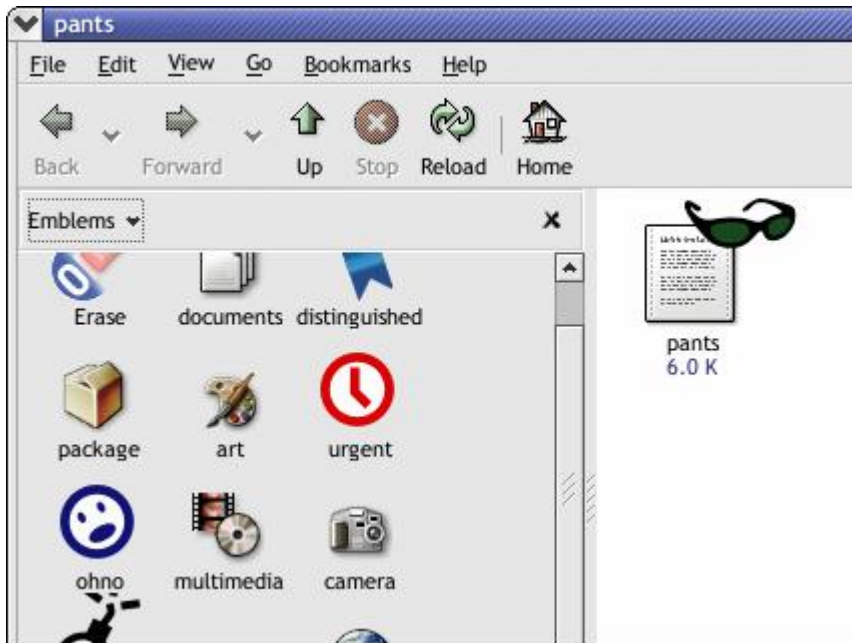


Figure 3. Emblems Screenshot

If, like me, you've accumulated hundreds of data files on your computer, within dozens of directories going down several levels, you'll appreciate Nautilus' bookmark feature. Simply browse your way to a directory you use often, click on the Bookmarks pull-down menu and select Add Bookmark. The next time you want to get to that directory, click on the icon created for it in the Bookmarks menu and you're there.

Configuring

Nautilus is also a graphical interface for configuring both GNOME and the underlying OS. You can reach the GNOME utilities and configuration programs through the menus or by clicking on the Nautilus pull-down menu labeled Go and choosing Start Here. A window then opens with four program group icons that read, Applications, Preferences, Server Settings and System Settings. Incidentally, the desktop had an icon for it in version 1.4, but it has been replaced by the configuration menus.

The Applications menu group connects to all the applications that appear in the main menu on the menu panel. Here you can launch programs or add

application launchers to the menus. However, this doesn't work in version 2.0, which was shipped with Red Hat 8.0. It has been fixed in version 2.2.

Under the Preferences menu group, you can modify a variety of settings such as the background, the default font and the mouse settings. You can pick a different theme or change the focus behavior of windows here. Those who are more keyboard- than mouse-prone will appreciate the Keyboard Shortcuts utility. With it you can create key combinations to do things like open favorite programs or switch workspaces. Many are already set up in this utility, but they can be modified.

The Server Settings program group provides links to utilities for configuring server applications such as Apache's web service. This program group will have more or fewer utilities depending on what's installed on a system. At a minimum, though, there is an interface to the xinetd services located in the / etc/rc.d/init.d directory. These are the same system services that are accessible from the old setup program.



Figure 4. Nautilus System Settings

The System Settings program group has all the good stuff for configuring a computer. Many of these utilities are coming from Red Hat but have been developed for GNOME. This includes an interface for date and time and a utility for changing the video display settings. Incidentally, this is now where you adjust your X configuration; it's no longer part of the setup program, in case you were frustrated not to find the X configurator there. With version 2.2, support for multiheaded display with multiple video cards and monitors was simplified. Waugh says, "Nautilus will manage both desktops with the same

process, and panels will be able to display on both heads, etc. There's even support for migrating applications between displays and such.”

Clicking on the Network icon in the System Settings window will open a utility (**neat**) for configuring network cards and the hosts file. The printer utility (**printconf-gui**) allows the user to add printers, set print drivers and restart the printer dæmon.

Accessibility

The improvement in GNOME 2 that will most directly affect increasing free software's desktop market share is accessibility for people with disabilities. “The US government can now use open-source desktop solutions, which wasn't going to happen otherwise because of government regulations”, says Pennington. He adds:

It also benefits normal users in a lot of ways. For instance, one of the accessibility requirements is full keyboard navigation [mentioned above]. You can do just about anything from the keyboard now. Also, themes have been enhanced because of accessibility regulations: color contrasts, default font size and so on.

With the accessibility barrier eliminated, Linux's target market has been expanded greatly, and options for all users have been improved in the process. Pennington says, “Sun Microsystems first brought accessibility to GNOME's attention. It was a huge project involving a couple years of work and about 20 developers. We're very excited about it and we're proud of what has been accomplished.”

Menus and Applications

Besides the system configuration menus mentioned above, in the main menu there are several other menus for launching applications. A menu labeled Accessories includes a calculator, a dictionary and a simple text editor (**gedit**). Under the menu labeled Office, OpenOffice is included along with Dia for creating organizational charts and flowcharts, along with MrProject for project management. The Graphics menu provides links to The GIMP and a few other image manipulation programs. The Games menu includes many games, some of which utilize the GNOME libraries: a couple solitaire games, a few popular line-up-the-dots games, a GNOME version of *Minefield*, *Mahjongg* and several others. I can't list all of the applications installed by default, but as you can see, streamlining the GNOME menus did not short-change the user.

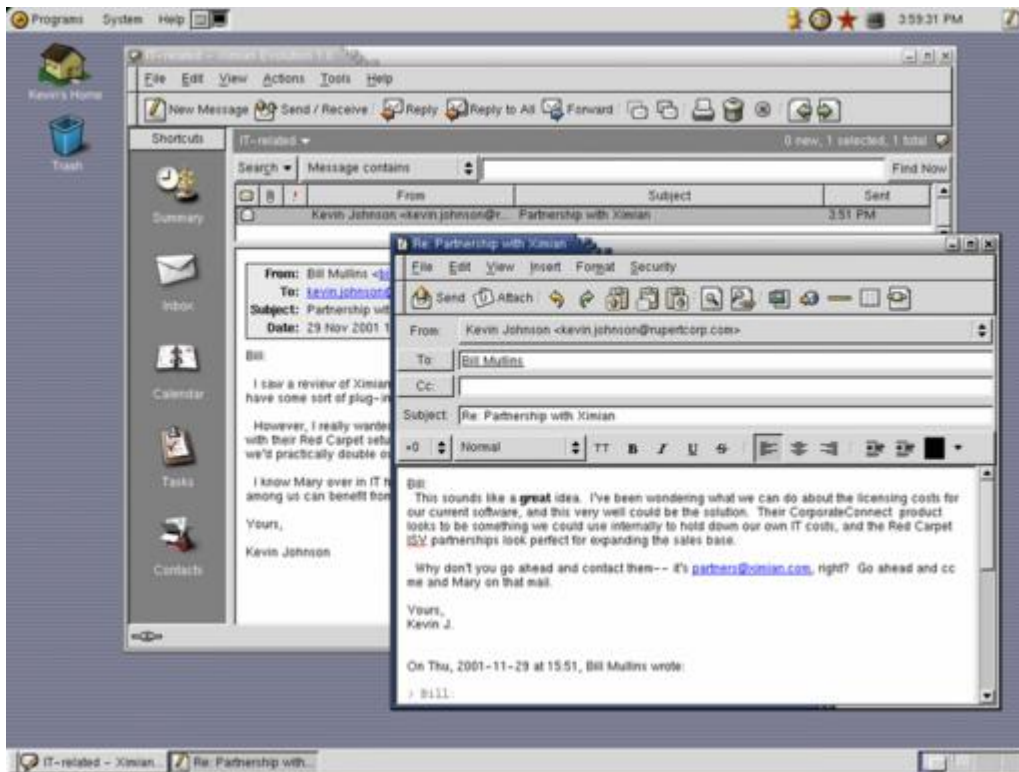


Figure 5. Ximian Evolution

Ximian Evolution

Probably the best application that makes use of the GNOME libraries is Ximian's e-mail client, Evolution. It was rated by *Linux Journal* in November 2002 as one of readers' favorite e-mail clients available for Linux. Evolution allows for multiple POP and IMAP accounts. It comes with e-mail filtering, spell checking and the ability to attach binary files. Although it works with standard POP and IMAP servers as is, with the addition of Ximian's proprietary Connector, users can connect to a Microsoft Exchange server for group address books and appointment planning—an important compatibility component.

One feature of Evolution that other e-mail programs don't have is virtual folders. "V-Folders allow the user more flexibility and ease of organizing e-mails—they're contextual views of your messages. That is something completely unique to Evolution", says Christine McLellan, senior product manager for Evolution. For example, in addition to a view of e-mail shown in the inbox in which messages can be sorted by date, subject or sender, a virtual folder is provided that shows only the unread messages. This virtual folder can make working through new messages quick and easy. Virtual folders also can be set up for messages with certain subjects or from certain people. A word of caution though, if you delete a message in a virtual folder, it's deleted from all folders at the same time.

Conclusion

The GNOME Foundation has made fabulous improvements to the desktop, and they have achieved much-needed consistency and stability with GNOME 2. “None of this would have happened without our developers, hundreds of which aren't paid by GNOME and are not sponsored by their employers”, says Ney. At this point, the Foundation's plans are to release stable updates every six months, with the next one (v. 2.4) scheduled for June 2003. Version 2.4 will include a Nautilus drag-and-drop CD burner function and more improvements to fonts. With their commitment to scheduled improvements to the desktop, GNOME has become a desktop environment that can be relied on by businesses, users and developers.



Russell Dyer is a Perl programmer and a MySQL developer living and working on a consulting basis in the New Orleans area. His Bachelor's degree is in English, and he also has been working on a Master's degree in English. He welcomes reader responses to his articles and can be reached at russell@dyerhouse.com.

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

[Advanced search](#)

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

[Advanced search](#)

Hacking Red Hat Kickstart

Brett Schwarz

Issue #108, April 2003

How to create a single CD for fast and easy customized installation.

Installing Linux is a relatively easy task. However, I was faced with the task of installing it on multiple machines repeatedly, which is time consuming and prone to errors. This problem affected our whole group and other groups that relied on us. So I started using Red Hat Kickstart to automate the installs. This helped, but there still was room for improvement. What I ultimately wanted was an automated installation that would fit on one CD, dynamically partition the hard drives and contain all of the updated packages. I wanted to be able to start an installation then walk away from the machine, returning only when it was done. To accomplish this, I supplemented Kickstart with a customized version of the Red Hat installation program, Anaconda.

Although not officially supported, Red Hat has made available several tools and documentation to assist in customizing an installation. I describe a few of the possible ways to do this here, which should give the reader enough information to get started.

The following topics are covered in this article: replacing packages with updates, reducing the installation size to fit on one CD, utilizing Kickstart in the custom installation and creating a custom message screen.

The reader should have a good understanding of Linux installations in general. I also assume that no esoteric hardware is being used, as other customizations may be needed to accommodate such hardware.

Setting Up the Build Machine

The first step is to prepare the build computer. Because the installer tools are specific to a particular release, the build computer needs to have the same Red Hat release as the one used on the target(s). For our example, Red Hat 8.0 is

being used. There are some differences between Red Hat 8.0 and previous versions, and you need to investigate them if you use a previous version.

Once the build computer has the correct release installed, the Anaconda packages need to be installed. These usually are not installed by default, so they need to be manually installed. They are located on the second CD of the standard Red Hat distribution and are named `anaconda`, `anaconda-runtime` and `anaconda-help` (optional).

The next step is to create a directory structure where the installation files will be located. The partition should have adequate space available, approximately 3GB. The actual location is based on your preference; for this article, the base directory is located at `/RH80`. Under this directory, we create directories for each of the CDs:

```
mkdir -p /RH80/CD{1,2,3}
```

We are not concerned with the source packages, so CD4 and CD5 are not included.

We make an additional directory where we can create the custom installation:

```
mkdir /RH80/ONE_CD
```

Now we can copy the contents of the CDs to the respective directories. Mount the first CD, then issue this command:

```
cp -a /mnt/cdrom/* /RH80/CD1/
```

Repeat this step for CD2 and CD3.

Copy the contents of the CD directories to the `ONE_CD` directory, but hard link them instead of actually copying the files. This saves space and is quicker:

```
cd /RH80
cp -al CD1/* ONE_CD/
cp -al CD{2,3}/RedHat/RPMS/* ONE_CD/RedHat/RPMS/
```

You'll get an overwrite `TRANS.TBL` message; you can answer `no`.

Selecting the Packages

Next we trim down the contents of the `ONE_CD` directory so it fits on one CD. I assume the CD size to be 700MB. I will not go into detail on how to do this, as

the list of files to remove from the distribution is different from one installation to another. However, here are some tips for trimming down the distribution:

- Don't include the source RPMs.
- Remove the dosutils directory, as these will be automated installs.
- Remove any unnecessary packages. This can be complicated, because you need to make sure that the dependencies are still intact.

You should keep a record of the files that were removed. You can use this list in case you need to back out, and you will need it later if you edit the comps.xml file.

For the package selection, I logged to a file all of the base and core group packages with their dependencies (according to the comps.xml file). In order to find this information, I used the script `getGroupPkgs.py` (see Resources):

```
cd /RH80/ONE_CD/RedHat/base
getGroupPkgs.py comps.xml > /RH80/pkglist
```

Any additional package names can be appended to the end of this file. After my list was complete, I removed the packages not on the list by using the `syncRpms.py` script (see Resources). The arguments are the package directory and the list of packages is generated from `getGroupPkgs.py`. This script removes the packages not listed in the package list and prints out the package names. We redirect that to a file so we have a log:

```
cd /RH80
syncRpms.py ONE_CD/RedHat/RPMS/ pkglist > pkgs_rem
```

We can monitor the installation size by using the `du` command. The `-h` option displays the result in human readable format, and the `-s` option gives a summary of the whole directory tree:

```
du -hs /RH80/ONE_CD
```

The `hdlist` files actually decrease in size after they are regenerated (see below), because we removed many of the packages. This in turn reduces the size of the CD image.

The tricky part about removing packages is they may break dependencies. Even though `getGroupPkgs.py` resolved the dependencies based on the `comps.xml` file, they are not guaranteed to be accurate. Adding additional packages may break dependencies as well. One way to verify their accuracy is to create a temporary RPM database, and then do a test install on that database with the packages you have selected:

```
cd /RH80/ONE_CD/RedHat/RPMS
mkdir /tmp/testdb
```



```
rpm --initdb --dbpath /tmp/testdb
rpm --test --dbpath /tmp/testdb -Uvh *.rpm
```

Look for any error messages regarding failed dependencies. If any appear, resolve the dependencies by either adding or removing files that caused the discrepancy, and repeat the above test.

Once the package dependencies have been resolved, you can download the package updates pertinent for your installation. Put these files under a separate directory:

```
mkdir -p /RH80/updates/RPMS/
```

Remove the old files from the build directory and then link the updated files to the build directory. Do this for each updated package (where *old_rpmfile* is the previous version of the package):

```
cd /RH80/ONE_CD/RedHat/RPMS/
rm
# ... remove each old rpm
cd /RH80/updates/RPMS/
cp -l
# ... do this for each rpm
```

You should keep a record of the updated packages, in case you need to back them out. It's also a good idea to check the dependencies and size one more time, in case they changed after the packages were updated.

Next, we check the internal checksum of each package with the `-K` option to `rpm`. First we need to import the key:

```
cd /RH80/ONE_CD/RedHat/RPMS
rpm --import /usr/share/rhn/RPM-GPG-KEY
rpm -K *.rpm | grep "NOT *OK"
```

This isn't strictly necessary, but because we downloaded package updates, this verifies they are valid.

Preparing the Installation Files

Once all of the packages have been updated, we need to regenerate the `hdlist` files. They contain only the headers of the packages, which allows Anaconda to retrieve the header information more quickly. Because we updated packages, we need to regenerate these files with the `genhdlist` tool, which is part of the `anaconda-runtime` package:

```
/usr/lib/anaconda-runtime/genhdlist /RH80/ONE_CD/
```

Next we need to handle the `comps.xml` file. This file defines package groups and package dependencies (although they are not guaranteed). The file

structure was changed in Red Hat 8.0 to be XML-based; in previous releases it was only a simple text file with some obscure tags. We need to ensure that packages defined within groups are included in our installation. We need to be concerned only with groups we are installing. If packages are missing (or if packages were added), we need to edit the comps.xml file (see Resources). Because we chose all of the packages in the Core and Base groups, however, we don't need to edit this file. We simply need to specify those groups under the %packages directive in the Kickstart file. See Listing 1 for an excerpt from the Kickstart file.

Listing 1. Excerpt from the Kickstart File

Technically, we can leave out the @Core and @Base groups, as they are installed by default. They are included here for illustrative purposes.

Creating a Custom Message Screen

We also want to create a custom message screen to give the user special instructions. The message screens are kept in the boot.img file (for CD-ROM installs) under the images directory. This is a DOS filesystem, so we can mount it to get to the contents:

```
cd /RH80/ONE_CD/images
mount -o loop -t msdos boot.img /mnt/boot
```

Looking in /mnt/boot, you see six message files: boot.msg, options.msg, general.msg, param.msg, rescue.msg and snake.msg. We create our own message file and call it custom.msg, an arbitrary name. Because snake.msg isn't really used, we replace that entry within syslinux.cfg with custom.msg. Edit syslinux.cfg in /mnt/boot and replace F7 snake.msg with F7 custom.msg.

A few other modifications were made to the syslinux.cfg file; refer to Listing 2. The default entry was changed from linux to ks. If the timeout occurs or if the user presses Enter at the boot prompt, then the ks label is used. Additionally, the timeout value was decreased from 600 to 60, so the installation can start sooner if there is no input from the user. The display entry was changed as well. Instead of boot.msg being the initial message screen, we wanted our custom message to be displayed. For the append line under the ks label, we added two things. The first is the keyword text, to enable text-based installs. Then we changed the keyword ks to ks=cdrom:/ks.cfg. This hard codes the Kickstart location so the user doesn't have to specify it at the boot prompt.

Listing 2. Modified syslinux.cfd File

Next we create our custom.msg file. Listing 3 shows our custom.msg. The contents of the file can be marked up, such as adding color around words. For

example, `^O09Custom^O02` changes the color of Custom to blue, and `^O02` changes it back. See the `syslinux` reference in the Resources section for more information.

Listing 3. Custom Message

Once we have composed the custom message, we need to umount the boot image:

```
umount /mnt/boot
```

Building the CD

Before actually creating the CD, you may want to test it by doing a network install. See the Red Hat Kickstart documentation on how to do this.

We want this custom installation to be automated, so we put the Kickstart file on the CD itself. You can create the Kickstart file with any text editor, or you can use the GUI tool called Kickstart Configurator.

In the `%pre` section, add a shell script to probe the hard drives and dynamically create the partition information based on the number of drives. We take advantage of the fact that Kickstart executes the `%pre` section and then re-reads the Kickstart file. When it reads it for the second time, it includes the `/tmp/partinfo` file where the `%include` directive is located (see Listing 1). The `/tmp/partinfo` file is the output from the script. We use the `list-harddrives` command, which lists the available hard drives and their sizes. Having the partition created dynamically frees us from having to create multiple Kickstart files that hard code the partition information.

Once the Kickstart file is created, name it `ks.cfg` and place it in the root directory of our installation tree (`/RH80/ONE_CD/`). It is possible to create more than one Kickstart file and place all of them on the CD. Different Kickstart files might address different hardware configurations.

We can now create the ISO image. From our previous steps, the distribution should be small enough to fit on one CD and contain all of the updated packages. The `mkisofs` program creates the image, and then we can copy the image to the CD. The command to create the ISO image is:

```
cd /RH80/ONE_CD
mkisofs -r -T -J \
-V "My Custom Installation CD" \
-b images/boot.img \
-c images/boot.cat \
-o /RH80/mydist.iso \
/RH80/ONE_CD
```

Refer to Table 1 for a description of the options.

Table 1. Options Used for mkisofs

The last parameter to `mkisofs` is the source directory of the contents that need to be included in the image file (e.g., our custom installation directory). Several other parameters are available that you may want to use. For example, the `-A`, `-P` and `-p` options add additional labeling information to the image. The `-m` and `-x` options also allow you to exclude certain directories and file patterns from the image. See the `mkisofs` man page for additional information.

Next, add a checksum to the ISO image. This is not strictly necessary, but it provides a way for end users to check the integrity of the CD. The tool to add a checksum to the ISO image is called `implantisomd5`. To add a checksum to the ISO image, use the following command:

```
implantisomd5 /RH80/mydist.iso
```

A companion tool, `checkisomd5`, can be used to verify the checksum for you:

```
checkisomd5 /RH80/mydist.iso
```

The CD also can be verified during the installation. After booting from the CD, the user can issue this command:

```
linux mediacheck
```

Now we can burn the image to the CD. I assume the CD writer is already set up on your system. We use `cdrecord` below, but you can use other programs as well. The command is invoked as:

```
cdrecord -v speed=4 dev=0,0,0 /RH80/mydist.iso
```

The `speed` and `dev` options depend on your system. The device for the `dev` argument can be determined by using the `-scanbus` option to `cdrecord`:

```
cdrecord -scanbus
```

Using the CD

Once the image is burned onto the CD, insert the CD into the target machine and boot the machine. You should get the custom message that you created earlier. At this point, you can either press `Enter` at the boot prompt or let it timeout. When it times out it uses the default label, which we specified as `ks` (Kickstart).

If we did everything right, the installation should proceed without user interaction. In my experience, the installation takes approximately ten minutes. This may differ depending on your exact configuration.

Conclusion

With a combination of Kickstart and a customized Anaconda, a powerful and flexible installation can be created. This installation greatly improved cycle time and reduced errors for my project. I was able to install multiple machines, multiple times almost effortlessly. In this article, I touched on only a few ways to take advantage of Kickstart and Anaconda, but there are many other possibilities. I encourage those interested to read the documentation in the Resources section and to join the Kickstart and Anaconda mailing lists for further information.

Resources



email: bschwarz@pamd.cig.mot.com

Brett Schwarz lives near Seattle, Washington, with his wife, son and dog. Although he is familiar with multiple platforms, his platform of choice is Linux. He is a computer and wireless systems consultant. He can be contacted via his home page at www.bschwarz.com.

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

[Advanced search](#)

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

[Advanced search](#)

The USB Serial Driver Layer, Part II

Greg Kroah-Hartman

Issue #108, April 2003

How can you create a USB device that works with the generic USB serial driver? Read and learn.

In the first part of this article [[LJ](#), February 2003], I introduced the USB serial layer and the basics of how to register a driver with the layer. This article explains some of the details about how data flows through the layer and how USB serial devices show up in sysfs.

Generic USB Serial Devices

In Part I of this article, I briefly mentioned the generic USB driver in the context of getting a USB device to communicate through it easily, with no custom kernel programming. Unfortunately, I didn't explain exactly how to do this, and many people wrote in with questions.

To create a USB device that works with the generic USB serial driver, all that is needed is two bulk USB endpoints on the device, one IN and one OUT. The generic USB serial driver will bind those two endpoints together into a single tty device that can be read from and written to from user space. For example, a device with the endpoints as described by `/proc/bus/usb/devices` (Figure 1) shows up as a single port device and produces the following kernel message when plugged in:

```
Generic converter detected
Generic converter now attached to ttyUSB0
(or usb/tts/0 for devfs)
```

Then any user can send data to the device through the `/dev/ttyUSB0` node.

```

T: Bus=03 Lev=02 Prnt=02 Port=01 Cnt=02 Dev#= 5 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=ffff ProdID=fff8 Rev= 0.01
C:* #Ifs= 1 Cfg#= 1 Atr=a0 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=00 Driver=serial
E: Ad=81(I) Atr=02(Bulk) MxPS= 16 Iv1=0ms
E: Ad=01(O) Atr=02(Bulk) MxPS= 16 Iv1=0ms

```

Figure 1. A Sample `/proc/bus/usb/devices` Entry

If a device has more than one bulk IN and bulk OUT pair, multiple ports are assigned to the device. For example, a device with the endpoints as described by `/proc/bus/usb/devices` (Figure 2) shows up as a two-port device and produces the following kernel messages when plugged in:

```

Generic converter detected
Generic converter now attached to ttyUSB0
(or usb/tts/0 for devfs)
Generic converter now attached to ttyUSB1
(or usb/tts/1 for devfs)

```

For this device, both `/dev/ttyUSB0` and `/dev/ttyUSB1` can be used to communicate.

```

T: Bus=03 Lev=02 Prnt=02 Port=01 Cnt=02 Dev#= 5 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=ffff ProdID=fff8 Rev= 0.01
C:* #Ifs= 1 Cfg#= 1 Atr=a0 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 4 Cls=ff(vend.) Sub=00 Prot=00 Driver=serial E:
Ad=81(I) Atr=02(Bulk) MxPS= 16 Iv1=0ms
E: Ad=01(O) Atr=02(Bulk) MxPS= 16 Iv1=0ms
E: Ad=82(I) Atr=02(Bulk) MxPS= 64 Iv1=0ms
E: Ad=02(O) Atr=02(Bulk) MxPS= 64 Iv1=0ms

```

Figure 2. Entry for a Two-Port Device in `/proc/bus/usb/devices`

The order of the endpoints is not important, so all of the IN endpoints could be first, followed by the OUT endpoints (unlike the previous examples that alternate). The USB serial core will take all of the IN and OUT endpoints and pair them up in the order they are seen. It also will assign an interrupt endpoint to a bulk pair, if one is present, but the interrupt endpoint will not be used by the generic driver; it can be used only by a USB serial driver within the kernel.

To get the generic USB serial driver to bind to the device, the USB vendor and product IDs need to be specified as a module parameter when the `usbserial` module is loaded. For example, to bind to the previously described device with a vendor ID of `ffff` and product ID of `fff8`, use the following command:

```

modprobe usbserial vendor=0xffff product=0xfff8

```

If the user cannot be expected to load the `usbserial` module with the specific device ID, or if more than one device ID should be used by the generic USB

serial driver, a very tiny driver can be written. An example of this is shown in Listing 1. In this driver, no callback functions are specified, only the product and vendor IDs of the devices that should be controlled. This is shown in the declaration of the struct `usb_serial_device_type`:

```
static struct usb_serial_device_type tiny_device = {
    .owner =          THIS_MODULE,
    .name =           "Tiny USB serial",
    .short_name =     "tiny",
    .id_table =       id_table,
    .num_interrupt_in = NUM_DONT_CARE,
    .num_bulk_in =    NUM_DONT_CARE,
    .num_bulk_out =   NUM_DONT_CARE,
    .num_ports =      1,
};
```

Specific vendor and product IDs should be listed in the `id_table` pointer:

```
static struct usb_device_id id_table [] = {
    { USB_DEVICE(MY_PRODUCT_ID, MY_DEVICE_ID1) },
    { USB_DEVICE(MY_PRODUCT_ID, MY_DEVICE_ID2) },
    { USB_DEVICE(MY_PRODUCT_ID, MY_DEVICE_ID3) },
    { } /* Terminating entry */
};
```

Listing 1. The Tiny Tiny USB Serial Driver

In all, this driver contains only two functions, which are two and three lines long, and three variable definitions. With it, all of the generic USB serial driver functionality will occur for the specified devices. The driver automatically will be loaded for the device when it is plugged in to the system, which is also a nice feature. This has to be one of the smallest working Linux kernel drivers possible. Compile it with:

```
echo "obj-m := tiny_tiny_usbserial.o" > Makefile
make -C <path/to/kernel/src> SUBDIRS=$PWD modules
```

The Windows operating system also supports this kind of device interface through the Windows USB OPOS serial driver, which will create virtual “COM” ports for the device. This allows hardware vendors to create USB devices that do not require any custom driver development for both Linux and Windows machines, which can be highly desirable.

Life Cycle of a USB Serial Device

When a USB-to-serial device is plugged in, a long series of steps are taken to allow a specific USB-to-serial driver to control an individual tty device. The steps are as follows:

- The USB hub driver detects a new device. It assigns a USB number to the device and reads the basic USB description from the device, which it then populates into a struct `usb_device` with a number of struct `usb_interfaces` that represent the whole USB device.

- The USB core takes the device and registers the USB interfaces with the kernel driver core.
- The kernel driver core looks through the currently registered list of USB drivers to determine if any of them will accept this device.
- Because this is a USB-to-serial device, the USB serial core accepts control of the device from the kernel driver core.
- The USB serial core builds up a single struct, `usb_serial`, and calls the specific USB serial driver's `probe()` function with this structure.
- The USB serial driver's `probe()` function initializes the device if it should and then returns control back to the USB serial core.
- The USB serial core creates the struct `usb_serial_port` structures depending on the number of serial ports on this specific device and then calls the USB serial driver's `attach()` function, if present.
- After the `attach()` function returns, the individual struct `usb_serial_port` structures are registered with the kernel driver core.
- The kernel driver core calls back into the USB serial core for every individual port.
- The USB serial core calls the individual `port_probe()` function in the USB serial driver for the port, if present, and then registers the port with the tty layer, completing the initialization process.

After this process, the tty device node is bound to the individual USB serial port. When the device node is opened by a user, the following steps happen in the kernel:

- The kernel looks up the device node and determines that the tty layer has registered this node, so it calls the tty layer's `open` function.
- The tty layer looks up the device and determines that the USB serial core has registered this node with it, so it calls `serial_open()` in the `drivers/usb/serial/usb-serial.c` file.
- The `serial_open()` function determines what specific USB serial driver is registered for this node.
- The module count for the specified USB serial driver is incremented in order to prevent it from being unloaded while a user is talking to the device.
- If the specified USB serial driver has an `open()` function, it is called with struct `usb_serial_port` for the specific port being passed to it.
- The USB serial driver then can do any hardware-specific open functionality that is needed and send off any USB urbs that are necessary to start accepting data from the device.

When a user calls `write()` on the device node to send data to the specified serial port, the following steps happen in the kernel:

- The kernel calls the `tty_write()` function within the tty core. It has previously set up this pointer during the open call, so it will not look it up again.
- `tty_write()` calls the line discipline's `write()` function for this specific tty device.
- The line discipline calls the USB serial core `serial_write()` function.
- The `serial_write()` function determines the specific USB serial driver used by this file and calls the `write()` function of it.
- The USB serial driver can then copy the data into a buffer and send it out the USB connection to the device, handling any special formatting issues the device might require.
- After the data has been sent completely, the driver can wake up the tty device in order to send any buffered data to it. This should be done with the simple call:

```
schedule_work(&port->work);
```

When data is received by the USB serial driver for a specific port, it should place the data into the specific tty structure assigned to that port's flip buffer:

```
for (i = 0; i < data_size; ++i) {  
    if (tty->flip.count >= TTY_FLIPBUF_SIZE)  
        tty_flip_buffer_push(tty);  
    tty_insert_flip_char(tty, data[i], 0);  
}  
tty_flip_buffer_push(tty);
```

When a user calls `read()` on the device node, any data in the tty flip buffer for this port is returned.

When the device node is closed by the user, the following steps occur within the kernel:

- The `tty_release()` function is called in the tty core by the kernel.
- `tty_release()` determines if this is the last reference held on this device node (remember, a device node can be opened by multiple programs at the same time). If it is, the USB serial core `serial_close()` function is called.
- The `serial_close()` function calls the USB serial driver's `close()` function, allowing it to shut down any pending USB transfers and get into a quiet state.
- The USB serial core then decrements the module count for the USB serial driver, possibly allowing it to be unloaded.

sysfs Representation of USB Serial Devices

In the previous description of how the USB serial device becomes bound to a specific USB serial driver, the kernel driver core is called a number of times. This happens because the USB serial core is represented as a bus within the kernel driver model, allowing multiple ports to be present on a single USB device.

For example, the following device is an eight-port USB-to-serial device on the first USB bus in the system. Its location in sysfs is `/sys/devices/pci0/00:09.0/usb1/1-1/1-1.1`. Within that directory are the following directories and files: `1-1.1:0/`, `bcdDevice`, `bConfigurationValue`, `bDeviceClass`, `bDeviceProtocol`, `bDeviceSubClass`, `bmAttributes`, `bMaxPower`, `bNumConfigurations`, `bNumInterfaces`, `idProduct`, `idVendor`, `manufacturer`, `name`, `power`, `product`, `serial`, `speed`, `ttyUSB0/`, `ttyUSB1/`, `ttyUSB2/`, `ttyUSB3/`, `ttyUSB4/`, `ttyUSB5/`, `ttyUSB6/` and `ttyUSB7/`.

The files in this directory provide the USB-specific information for this device, as do the files in the `1-1.1:0/` directory, which is the first interface on this device. The `ttyUSB*` directories are created by the USB serial core and contain the following files: `dev`, `name` and `power`.

The `dev` file contains the major and minor number for this specific device, which then can be used to determine the proper device node for talking to it. In the `/sys/bus/usb` directory, this USB device is seen as being bound to the `io_edgeport` USB driver (Figure 3).

```
/sys/bus/usb
|-- devices
|   |-- 1-0:0 -> ../../../../devices/pci0/00:09.0/usb1/1-0:0
|   |-- 1-1 -> ../../../../devices/pci0/00:09.0/usb1/1-1
|   |-- 1-1.1 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1
|   |-- 1-1.1:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/1-1.1:0
|   |-- 1-1.2 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.2
|   |-- 1-1.2:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.2/1-1.2:0
|   |-- 1-1:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1:0
|   |-- usb1 -> ../../../../devices/pci0/00:09.0/usb1
|-- drivers
|   |-- hub
|   |   |-- 1-0:0 -> ../../../../devices/pci0/00:09.0/usb1/1-0:0
|   |   |-- 1-1:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1:0
|   |-- io_edgeport
|   |   |-- 1-1.1:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/1-1.1:0
|   |-- p12303
|   |   |-- 1-1.2:0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.2/1-1.2:0
|   |-- usb
|   |   |-- 1-1 -> ../../../../devices/pci0/00:09.0/usb1/1-1
|   |   |-- 1-1.1 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1
|   |   |-- 1-1.2 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.2
|   |   |-- usb1 -> ../../../../devices/pci0/00:09.0/usb1
|   |-- usbfs
|   |-- usbserial
```

Figure 3. The /sys/bus/usb Tree

There is also a usb-serial bus, which shows the individual USB serial ports that are registered with the kernel (Figure 4). As these individual ports are tty devices, they also show up in the tty class directory (Figure 5).

```
/sys/bus/usb-serial/
|-- devices
|   |-- ttyUSB0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB0
|   |-- ttyUSB1 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB1
|   |-- ttyUSB2 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB2
|   |-- ttyUSB3 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB3
|   |-- ttyUSB4 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB4
|   |-- ttyUSB5 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB5
|   |-- ttyUSB6 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB6
|   |-- ttyUSB7 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB7
|-- drivers
|   |-- edgeport_1
|   |-- edgeport_2
|   |-- edgeport_4
|   |-- edgeport_8
|   |-- ttyUSB0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB0
|   |-- ttyUSB1 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB1
|   |-- ttyUSB2 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB2
|   |-- ttyUSB3 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB3
|   |-- ttyUSB4 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB4
|   |-- ttyUSB5 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB5
|   |-- ttyUSB6 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB6
|   |-- ttyUSB7 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB7
|-- generic
```

Figure 4. The /sys/bus/usb-serial Tree

```
/sys/class/tty/
|-- devices
|   |-- 0 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB0
|   |-- 1 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB1
|   |-- 2 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB2
|   |-- 3 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB3
|   |-- 4 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB4
|   |-- 5 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB5
|   |-- 6 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB6
|   |-- 7 -> ../../../../devices/pci0/00:09.0/usb1/1-1/1-1.1/ttyUSB7
|-- drivers
|   |-- usb-serial:edgeport_1 -> ../../../../bus/usb-serial/drivers/edgeport_1
|   |-- usb-serial:edgeport_2 -> ../../../../bus/usb-serial/drivers/edgeport_2
|   |-- usb-serial:edgeport_4 -> ../../../../bus/usb-serial/drivers/edgeport_4
|   |-- usb-serial:edgeport_8 -> ../../../../bus/usb-serial/drivers/edgeport_8
|   |-- usb-serial:generic -> ../../../../bus/usb-serial/drivers/generic
```

Figure 5. The /sys/class/tty Tree

Through all of these different links back to the single USB device, the type of USB device, how many tty ports it has and what type of USB serial driver controls it, easily can be determined. This is also much more information than what was shown in the /proc/tty/driver/usb-serial file, as described in Part I of this article.

The sysfs interface is described here only briefly, but it contains a wealth of information about all physical and virtual devices that are contained in a system at a given point in time. For a better description of sysfs and the kernel driver model, see Pat Mochel's 2003 linux.conf.au paper at www.kernel.org/pub/linux/kernel/people/mochel/doc/lca.

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

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

[Advanced search](#)

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

[Advanced search](#)

The Linux Kernel Cryptographic API

James Morris

Issue #108, April 2003

A new general framework offers much-needed crypto services to all parts of the kernel.

This article provides a brief overview of the new cryptographic API for the Linux kernel. It is aimed at anyone with a technical interest in Linux, such as system administrators, and other curious people who would like to gain insight into the API's design, implementation and application. Some knowledge of kernel internals is useful but not essential for a broad understanding of API concepts.

The history of this API is short. Not long before the Halloween 2002 kernel feature-freeze, an IPSec implementation being developed by Dave Miller and Alexey Kuznetsov became slated for inclusion into the 2.6 kernel. IPSec requires cryptographic support within the kernel, which along with an increasing general need for kernel cryptography, prompted the development of a new cryptographic API.

Design

Although initially aimed at supporting IPSec, the API has been designed as a general-purpose facility, with potential applications including encrypted files, encrypted filesystems, strong filesystem integrity, the random character device (/dev/random), network filesystem security (for example, CIFS) and other kernel networking services requiring cryptography.

A specific design requirement was that the API work directly in place on page vectors. A page is the primary unit of memory managed by the kernel. A page vector-based API allows for deep integration with kernel substructures, such as the VFS and networking stack, as well for as scatter-gather operations. In the case of IPSec, cryptographic transforms may be applied directly to discontinuous memory pages associated with network packets.

Simplicity was a significant design goal, which is always a good idea in general, and particularly important for kernel and security code.

Deployment flexibility was another goal. For example, the API has a flexible policy toward algorithms; they can be loaded dynamically as kernel modules, without the API needing to know anything about them in advance.

Future design goals include:

- Hardware support for cryptographic accelerator cards and NICs with IPsec offload.
- Support for specification of algorithm preferences when multiple implementations are available, for example, optimized assembler versions and various hardware implementations.
- Asymmetric cryptography support (RSA), which may be needed in the kernel to support multicast IPsec and kernel module signature verification. This may be a contentious issue, as asymmetric cryptography is generally slow and complicated—both are very good reasons to exclude it from the kernel.
- A unified API for user-space applications wishing to utilize available cryptographic hardware, such as SSL, IPsec key exchange, secure routing protocols and DNSsec.
- Further optimizing the API memory footprint to cater to embedded systems scenarios.

Algorithms

Three types of algorithms are currently supported by the API:

1) Digests (one-way hash functions)--these take arbitrary messages and produce short, fixed-length message digests. To be one-way, the hash function must be designed so it is easy to generate the hash but difficult to compute the original message from the hash. For cryptographic purposes, hash functions need to be collision-resistant, so that it is difficult for two messages to hash to the same value. Applications include ensuring data integrity and generating message authentication codes for network protocols. Examples of digest algorithms are MD5 and SHA1.

A message authentication scheme called HMAC (RFC2104) is included within the API, which will operate on any standard digest algorithm. This is currently used to generate authentication data for IPsec packets.

2) Ciphers—these algorithms implement symmetric key encryption, where a plain-text message is encrypted with a key to produce ciphertext. Generally, the

same key is used to decrypt the ciphertext back into the original plain text. It should be easy to encrypt and decrypt messages with the key (which must be kept secret) but difficult to do so without it. Applications include encrypting data to ensure privacy and generation of message authentication codes. Examples of cipher algorithms are Triple DES, Blowfish and AES.

There are two types of ciphers: block ciphers operate on fixed-length blocks of data (e.g., 16 bytes), and stream ciphers use a key stream to operate on as little as one bit of data at a time.

Ciphers also may operate in a variety of modes, such as Electronic Codebook (ECB), where each block of plain text is simply encrypted with the key, and Cipher Block Chaining (CBC), where the previously encrypted block is fed into the encryption of the next block.

3) Compression—this is often used in conjunction with encryption so that it is more difficult to exploit weaknesses related to the original plain text as well as to speed up encryption (i.e., compressed plain text is shorter). By definition, encrypted data should be difficult to compress, but this adversely affects performance over links that normally utilize compression. Compressing data before encryption helps reduce this performance hit in many cases. Examples of compression algorithms are LZS and Deflate.

So far, algorithm implementations from well-known sources have been adapted for use with the API, as they are more likely to have been reviewed and widely tested. For inclusion into the mainline kernel, algorithms generally must be patent-free (e.g., IDEA will not be a candidate for inclusion until around 2011), based on open, recognized standards and submitted with a set of test vectors.

Page Vectors

Before discussing the API structure, let's briefly look at memory pages and page vectors. As mentioned previously, a page is the fundamental unit of memory managed by the kernel (on i386, pages are 4KB in size). Consider a buffer containing, say, 1,460 bytes of user-space data. It belongs to a specific page in the kernel, offset from the start of the page by some amount, and has a length of 1,460 bytes. This buffer can be represented as a page-based tuple:

```
{ page, offset, length }
```

An interface, such as the cryptographic API that works directly with pages, needs to deal with this tuple, or page vector. An existing kernel data structure called a scatterlist is employed, which contains a page vector and normally is used for scatter-gather DMA operations.

The cryptographic API uses scatterlists to operate on arrays of discontinuous page vectors. The primary purpose of scatter-gather in the kernel is to avoid unnecessary copying of data. It also seems to result in cleaner code. Many readers will be familiar with scatter-gather I/O in the form of the `readv()` and `writev()` system calls. The kernel cryptographic API uses the same general concept but operates on pages instead of plain memory buffers.

API Structure

The API deals with two primary objects:

- Algorithm implementations—kernel modules that contain the underlying algorithm code.
- Transforms—objects that instantiate algorithms, manage internal state and handle common implementation logic. Transforms are managed by `crypto_alloc_tfm()` and `crypto_free_tfm()`. A set of API wrappers are provided to simplify transform use and to allow the properties of a transform's underlying algorithm to be queried.

The following pseudo-code demonstrates a typical use of the transform interface, where some kernel code needs to encrypt data using the Blowfish cipher in electronic codebook (ECB) mode:

```
tfm = crypto_alloc_tfm("blowfish",  
                      CRYPTO_TFM_MODE_ECB);  
crypto_cipher_setkey(tfm, key, keylength);  
crypto_cipher_encrypt(tfm, &scatterlist,  
                     numlists);  
crypto_free_tfm(tfm);
```

As shown in Figure 1, the API is layered so that core logic is hidden from cryptography users and algorithm implementors. This core logic includes generic transform management, scatterlist manipulation and abstraction of underlying algorithms. Further down, per-algorithm-type logic is handled, such as cipher processing modes and utilizing digests for generating message authentication codes.

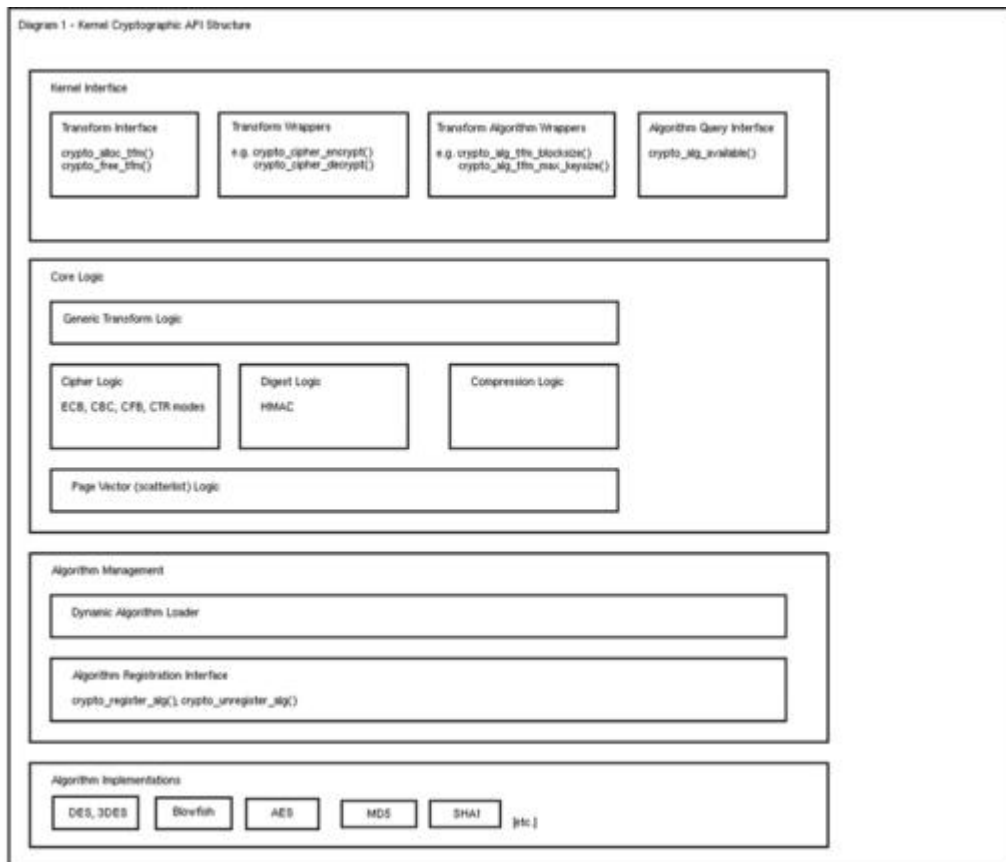


Figure 1. Structure of the Kernel Cryptographic API

The algorithm management layer contains logic for locating, loading and reference counting algorithm implementations. The latter is required to prevent nasty things from happening if an attempt is made to unload an algorithm module that is still in use.

An algorithm runtime query interface is provided so that calling code can determine which algorithms are available on the system. This is primarily intended for use by key negotiation protocols, such as ISAKMP/IKE.

Finally, the algorithm registration interface allows modules to register one or more algorithms, specifying various properties such as the name of the algorithm, its block size and minimum and maximum key sizes. The list of currently registered algorithms and their properties may be viewed in `/proc/crypto`.

Conclusions

This is still a young API that is likely to evolve somewhat, especially if some of the future design goals listed here are implemented.

In terms of API users, IPsec works and performs well, especially for a first cut with no performance optimizations. Existing kernel components that need

cryptography are expected to convert to the new API over time, and hopefully, cool new projects will be developed because of it.

Acknowledgements

Many thanks to David Miller and Nancy Chan for reviewing this article.

Resources

email: jmorris@intercode.com.au

James Morris is a software developer involved with the Netfilter, LSM, SELinux and Linux kernel cryptographic API projects. He works as an independent consultant in Sydney, Australia.

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

[Advanced search](#)

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

[Advanced search](#)

Content Management

Reuven M. Lerner

Issue #108, April 2003

Give your web site newspaper-grade content management with open-source software.

Remember the good ol' days of the Web? Back when a webmaster was a jack-of-all-trades, doing everything from graphic design to database programming to DNS table manipulation? As the Web matured, however, one-person web sites became increasingly rare. True, it's still relatively easy for a someone to create and maintain a simple web site, but even the smallest organizations typically split responsibility between programmers, designers and the people who provide content. Moreover, many organizations want different people to be responsible for different types of content, with each having ultimate authority over a particular section.

Of course, this is old hat to the publishing world. Back when I edited my college newspaper, we used a composition and typesetting system called Atex. Atex was beloved for many reasons but mostly because it worked the way that newspapers do. Reporters using an Atex system would send articles to their editors by pressing the Send button on a massive, specialized keyboard. Editors could look at the list of articles awaiting their attention, edit articles, send an article back to the reporter who wrote it or send articles onto the Typesetting department. By design, everyone was forbidden from viewing, modifying or retrieving articles they had sent to the next person in the process chain. The image of a reporter shouting "stop the presses!" might be romantic and inspiring, but it is also unrealistic in today's world, where newspapers are businesses with tight deadlines.

As web sites grow to resemble newspapers, we should not be surprised to see them adopting software—known as content management systems, or CMS—that works much like Atex used to do. But organizing documents, people and work flow is a difficult task, particularly if you try to put everyone's needs into a single software package. So even though content management is crucial to an

increasing number of web sites, CMS salespeople have gained a reputation for selling bloated, expensive software that is large on promises and small on delivery.

What Does a CMS Do?

One of the problems with content management is that every web site has different needs. For this reason, proprietary CMS software usually is sold in two parts. The customer first pays for the basic software and then pays at least as much in consulting and support services. Thus, CMS software is not only expensive but requires a fair amount of implementation and testing time. In other words, a CMS usually is closer to a toolkit than a finished application. Most of these toolkits include the following functionality:

- **Users:** if everyone on a web site is going to be given different permissions, obviously each user will need a different login. A CMS thus comes with user-management software, allowing you to create, delete, edit and ban users on the system. Most systems also make it possible for users to retrieve forgotten passwords.
- **Permissions:** just as Linux allows you to set read, write and execute permissions on different files, CMS software typically allows the site administrator to define different permissions for each user on the system. To return to our newspaper analogy, reporters are allowed to enter content, editors can modify content (or return it to a reporter) and publishers can make content publicly available (or return it to an editor).
- **Groups:** although you theoretically can assign permissions to individual users, this quickly becomes tedious. So, most CMS software allows you to group users together for the sake of assigning permissions. For example, you can indicate that Tom, Dick and Harry can write and edit but not publish, or you can assign these permissions to the Canonical Names group, with the same effect.
- **Templates:** many templating systems exist, including JSP, HTML::Mason and PHP. The best ones separate design, content and programming logic from each other, so designers, writers and programmers can work on a site simultaneously without stepping on one another's toes.
- **Publishing:** the Web's biggest double-edged sword is its instantaneousness. The moment you modify foo.html on your server, everyone can see what changes you made. What if you made a mistake? What if you want to test the file beforehand? The CMS solution is to mark each piece of content as published when it should be viewed by the outside world. Until an article has been published, it is invisible.
- **Staging or previewing:** just as newspaper and magazine publishers want to see what the finished product will look like before they begin to print actual copies, web publishers want to preview their site before it is live on

the Web. Thus, many sites run staging servers, identical in most ways to their production servers except they are hidden from the outside world. Testing is done on these preview servers; when the editor or publisher is satisfied, content is pushed to the production servers. A CMS almost certainly will allow you to set up your system in this way.

- **Work flow:** staging is the final step in what might be a long journey from an author's workstation to a production web server. How content makes its way through the system is known as work flow, and much of what a CMS does is allow you to define and manage that work flow. Should reporters be allowed to yank stories back from their editors? How many levels of editors do you want? Where do designers fit in? Who gives the final send-off to content? All of these questions are handled by the work-flow portion of a CMS.
- **Publishing dates:** the good news about the Web is that things are published instantaneously. But what if your corporation is announcing a stock split and cannot reveal that information until 9:00 **AM** on Monday? You could sit next to the computer, waiting until the clock strikes 9:00 to press the Enter key and revealing the document for everyone to see. Or you can use a CMS, which typically allows you to specify when an article will appear, as well as when it should expire.
- **Web-based editing:** although a web browser is one of the worst possible programs to use for serious text editing, most CMS systems allow you to write some or all of your documents using your browser. To be fair, just about every CMS also lets you upload files from your local computer. Web-based editing comes in handy when you're on the run or want to touch up one or two things. Of course, any CMS that offers such editing facilities also checks that someone trying to edit a page is authorized to do so.
- **Search:** most CMS packages offer some sort of search facility, so you can find documents within the system.

Although this list is by no means exhaustive, it should give you a sense of the types of problems that a CMS tries to solve. But as you can imagine, every CMS offers a slightly different set of features and different ways of attacking these problems.

Because a CMS spends much of its time storing, retrieving and tracking content, it should come as no surprise that a database is almost essential to a CMS. Commercial CMS packages typically expect you to use a proprietary database system, such as Oracle or Microsoft's SQL Server. As you might expect, open-source CMS software generally is designed to work best with open-source databases, such as MySQL or PostgreSQL. Zope's Content Management Framework (CMF), which is a toolkit for creating a custom CMS, also uses a

database, but in this case, it's the built-in Zope Object Database (ZODB) rather than an external relational database.

Content Management vs. Application Development

If you have ever developed serious web applications, you immediately will see a large degree of overlap between the features a CMS offers and the features you expect from a web application server. Most CMS software sits on top of a web application server, using its underlying infrastructure to handle HTTP connectivity, users, groups, permissions and even the database API. In some ways, CMS was the first popular class of application to be deployed on the Web, much as spreadsheets were the first applications used on personal computers.

Overall, it's a good thing CMS software is written on top of an application server, especially in the open-source world. This means you can add new modules to the core CMS, handle new types of documents, change the templates, extend the database and add new types of permissions and workflow rules. But it's important to remember the difference between an application server and a CMS. The former provides the infrastructure for creating applications, and the latter is an application you can customize.

So if you're looking to create a web-based newspaper, magazine or corporate news site, a CMS is undoubtedly the right type of software for you. But if you want to create a web-based application that tracks donations to your favorite charity, a CMS probably won't provide the flexibility you need. The difference between web applications and web publications has always been a murky one, but as web applications become increasingly sophisticated, CMS software will be recognized as one type of product you can run on a web platform.

Because content management systems normally run on top of an application server, your choice of CMS might depend on the type of server on which it runs. Many companies have moved to J2EE (Java 2 Enterprise Edition) as their underlying platform. Indeed, the well-known Vignette CMS originally was designed to work with Tcl but migrated to J2EE when the buzz surrounding J2EE became too great to ignore. Because J2EE is a standard, rather than a product, customers can choose application servers and CMS software separately. You can use the open-source Tomcat/JBoss duo or the proprietary offerings from companies like BEA or IBM.

If you dislike Java, or if your development team is more familiar with another set of technologies, you might consider a non-J2EE CMS. Such products do exist, and we will look at several of them in the coming months, such as Zope's CMF, the CMF-based Plone, Bricolage (Perl/PostgreSQL), PHPNuke/PostNuke/Xoops (PHP) and Midgard (PHP).

Regardless of what technology you decide to use, a CMS is increasingly necessary and useful for producing web sites. Even if you're the only person working on your web site, moving to a CMS is probably a wise move, if only to help standardize the look, feel and delivery of content on your site. And, if you ever decide to add new types of content, the CMS will probably be able to handle it, though you might need to tinker with it somewhat.

Conclusion

CMS software is probably the first type of application designed for the Web. Most content management solutions are expensive and proprietary, but an increasing number of open-source options are available for those who want greater freedom and lower cost. Given that content management systems normally need a great deal of customizing and tuning, this is another niche for which open-source tools are an excellent fit.

Resources

Reuven M. Lerner (reuven@lerner.co.il) is a consultant specializing in open-source web/database technologies. He and his wife Shira recently celebrated the birth of their second daughter, Shikma Bruria. Reuven's book *Core Perl* was published by Prentice Hall in early 2002, and a second book about open-source web technologies will be published by Apress in 2003.

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

[Advanced search](#)

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

Sometimes, You Have to Do It Yourself

Marcel Gagné

Issue #108, April 2003

How to set up and use KDE Desktop Sharing and redesktop, the remote-control package.

François, make sure that hub is plugged in. *Merci, mon ami.* Now watch as I connect to the terminal in the wine cellar's east wing. You see, I have full access from up here. I can control the desktop, start applications, install software and so on. In fact, François, if you were down there now, we could both use the desktop. What do you mean, "But what is it good for?"

Sometimes, François, there is no control like remote control. If you could share your Linux desktop with another user or take control of another user's desktop as needed, you could save a great deal of time controlling the session or simply observing what is happening. For instance, let's pretend a certain waiter recently installed his new Linux system and occasionally needs a little guidance. *Mais non*, François, it is purely a hypothetical question. Your expertise with the desktop is well known. Please do not fret.

Ah, *mes amis!* Welcome to Chez Marcel, home of tantalizing Linux fare, great service and fine wine. Please sit and make yourselves comfortable. François, why don't you run down to the cellar? Something Italian today, I think. Bring back the 1997 Brunello di Montalcino for our guests.

I was explaining to François that there are times when nothing works better than taking control of a remote desktop in order to do what needs to be done. Perhaps the best incentive is the office environment. Using desktop sharing, a system administrator could deal with individual desktop issues without leaving the comfort (or convenience) of the office. Do you need to show a user how to add an icon to the desktop? Connect to their desktops and have them watch. Have you received a call asking for help interpreting an error message? Connect to the system and ask the user to recreate the scenario while you watch. The possibilities are endless.

An excellent cross-platform remote-control package is VNC from AT&T Laboratories in Cambridge, England. The problem with VNC is it doesn't provide any means of controlling the main X display (**\$DISPLAY:0**), so taking control of a user's desktop to fix a problem or to show them how to do something is out of the question.

Enter KDE Desktop Sharing from creator Tim Jansen. The package is scheduled to be part of the standard network offerings in KDE 3.1, which may be out as we speak. Those running KDE 3.0 also can take advantage of this great package by visiting Tim's site, www.tjansen.de/krfb, and downloading the latest source.

Building the package is the standard extract and build five-step with a twist. The twist is the package is stored as a bzip2 file rather than as a straight gzip file.

```
tar -xjvf desktopsharing-0.7.tar.bz2
cd desktopsharing-0.7
./configure --prefix=
make
su -c "make install"
```

The reason I suggest adding the prefix shown above is KDE Desktop Sharing becomes a KDE control center module, so letting your KDE implementation know where the software lives is a good idea. For instance, I've installed this on Red Hat, Mandrake and SuSE systems. Red Hat and Mandrake's KDE directories are subdirectories of /usr, while SuSE's are in /opt/kde3.

Before we move on to using the product, let me clear up one other potential area of confusion. Different distributions sometimes opt for nonstandard KDE menus, meaning that things aren't where this little installer expects to find them. A case in point is the Mandrake 9.0 desktop. Despite the installation having gone smoothly, the Desktop Sharing configuration option did not appear in the Control Center menu. The fix is an easy one, once you know about it. Here is what you do: from the desktopsharing distribution directory, change to the krfb/kcm_krfb directory and manually install the desktop entry for the KDE Control Center (the following is one single command line):

```
/usr/bin/install -c -p -m 644 kcmkrfb.desktop
/usr/share/applnk-mdk/Configuration/KDE/Network/
```

Once again, I stress that when this feature becomes part of the actual KDE distribution (release 3.1), installation won't be a problem at all. Now, when you have finished compiling, installing and tweaking menus, make sure to log out from your KDE desktop before you continue. While the desktop reboots, why not take a moment to appreciate the bouquet from this most excellent wine, *non?*

To configure your client PC for remote access, start by bringing up the KDE Control Center from your application starter menus (the big K in the lower left-hand corner). You also can start the control center by typing **kcontrol &** at a shell prompt. When kcontrol starts, click on the Network icon in the left-hand sidebar menu and select Desktop Sharing, as shown in Figure 1. As you can see, two tabbed windows appear on the right-hand side. One is labeled Access and the other is Network.

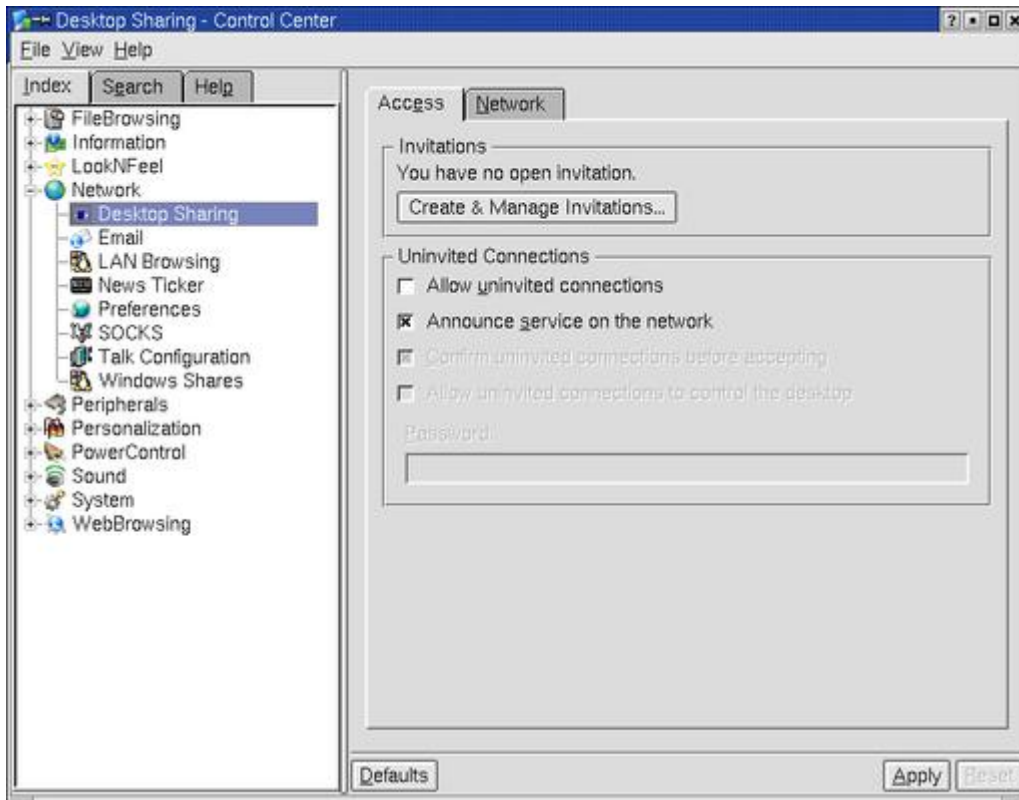


Figure 1. Configuring KDE Desktop Sharing from the Control Center

The Network tab takes the least amount of explanation, so I will cover it first. Click the tab, and you'll have the opportunity to override the default to Assign port automatically. KDE Desktop Sharing's default port is 5900, but unchecking this box here makes it possible to assign a specific port number.

On to the Access tab. Invitations are the means by which access is granted to the desktop, but it also is possible to allow uninvited connections as well. I suspect, *mes amis*, that I do not have to explain the security implications of this course of action. For this reason, allow me to tell you how to create and manage invitations.

For starters, click the button that says Create & Manage Invitations. The window that pops up offers you two important choices. You can create either a New Personal Invitation or a New E-mail Invitation. Let's start with a personal invitation.

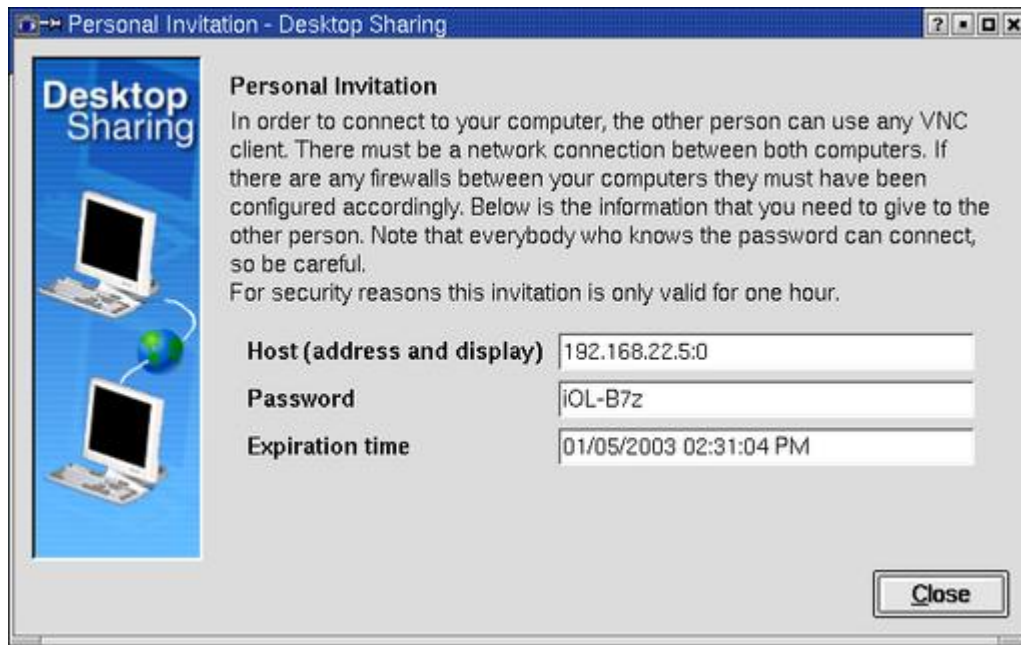


Figure 2. Creating a Personal Invitation for a KDE Control Session

For security reasons, the invitation itself lasts for only an hour. If you don't do anything else, Desktop Sharing automatically comes up with a password and an expiration time for the session. The host address necessary for the connection also is displayed. Overriding either the password or the expiration time is not allowed. Make sure you pass on the information as it is shown to the person who will be connecting. When you have passed on the information (or written it down), click Close.

The other option is an e-mail invitation. The only catch here is you are sending the means to access your system via e-mail during that one hour period. If you choose this option, you'll receive a warning about plain-text e-mail over the Internet and the *wisdom* of encrypting said e-mail. Click Continue to get past the warning and a KMail message appears, ready for you to click Send. If no one answers the invitation, it disappears within an hour. Incidentally, you also can manage invitations with the following command:

```
krfb &
```

Before we move on, click Close to get past all those invitations, and we'll have another look at the second means of providing access, uninvited connections. If sending an e-mail invitation presents interesting security concerns, then a wide-open, permanent invitation should ring additional bells. That said, in an office environment it also may be the sanest method of giving yourself access. If you check on Allow uninvited connections, you still have to assign a password for connecting. Furthermore, you have the opportunity to Confirm uninvited connections before accepting. You also can decide to give those uninvited connections the ability to control the desktop.

To connect to the desktop, you can use any VNC client; but the slicker way is to install KDE Desktop Sharing on the controlling desktop also. You'll find the interface friendly and appealing. To start the client, either select it from the Internet menu under the big K or call the program directly from the command line:

```
krdc &
```

After creating an open invitation with a confirm option, a remote client trying to connect generates a warning message asking whether you want to allow that connection (Figure 3).

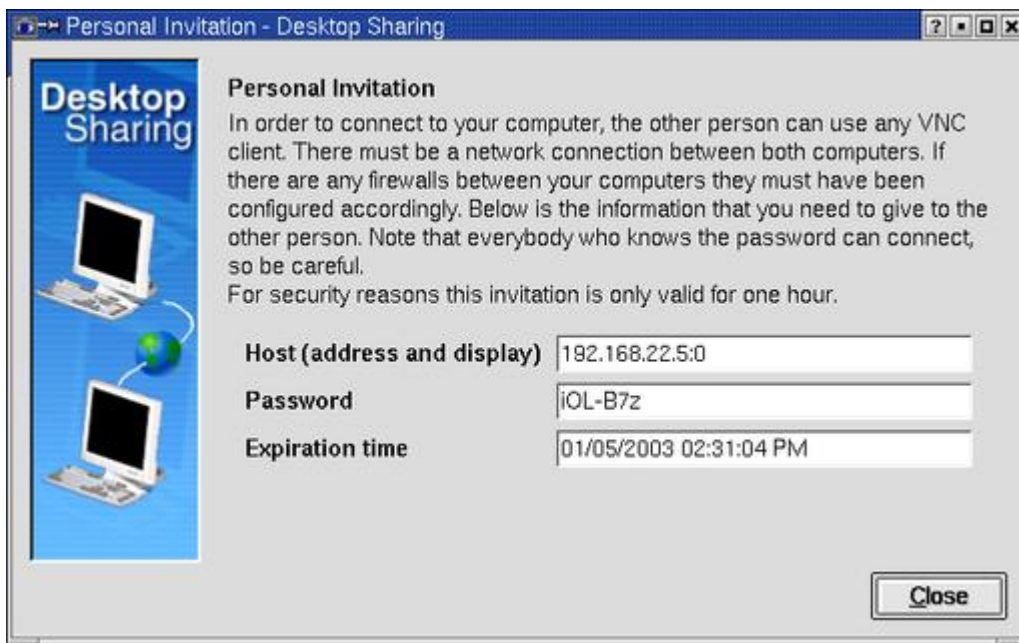


Figure 3. A Request for Desktop Control

After accepting the request, the remote user still has to enter the password, at which point you'll see a nice blue eye staring at you from the system tray.

One of the great things about this particular program is you can resize or scale the virtual desktop to almost any size. Size your window, then click on the magnifying glass icon. If you would like a particularly psychedelic experience, set up an invitation on your own machine and try connecting to it. You'll receive an endless cascade of desktops, an effect much like standing between two parallel mirrors facing each other (Figure 4).

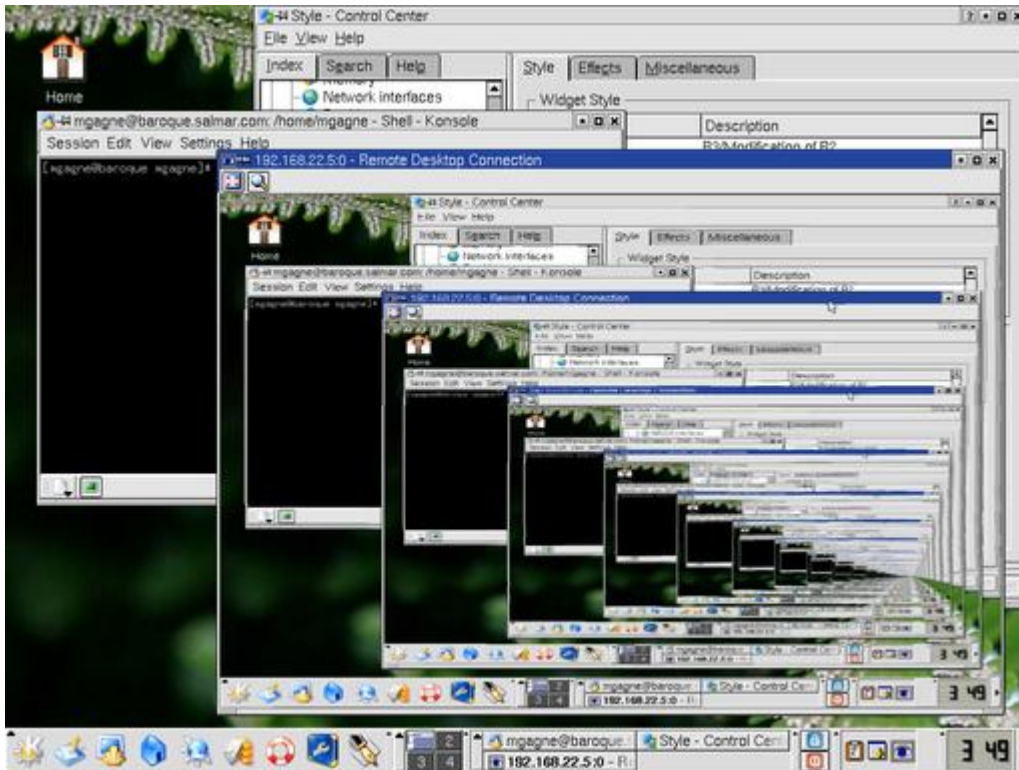


Figure 4. Fun-House Mirrors, Linux Style!

Ah, it is very exciting to consider how much closer Linux can bring us, *non*? It is true even when dealing with non-Linux systems. Allow me to demonstrate.

Another remote-control package worth your consideration is a little something Matt Chapman has cooked up called `rdesktop`. Here's the idea. From time to time, you may have to work on a box running Windows 2000. If that box requires you to connect using Win2K's Terminal Server, you no longer need to shut down your Linux system to do your work.

Simply put, `rdesktop` is a GPLed Windows (NT/2000) Terminal Server client, which means it uses RDP (remote desktop protocol). If you'd like to use `rdesktop` and keep running a Linux desktop in the process, pick up your copy of the source at www.rdesktop.org. Build it using the classic extract and build five-step:

```
tar -xzf rdesktop-1.1.0.tar.gz
cd rdesktop-1.1.0
./configure
make
su -c "make install"
```

The whole process should take no more than a few seconds.

When the installation is complete, you can start the program like this:

```
rdesktop -u Administrator -p PaSsWoRd 192.168.22.212
```

The -u parameter specifies a user account on the Windows server, while the -p option specifies the password. Have a look at Figure 5 for a screenshot of my KDE 3.1 desktop running rdesktop to a remote Windows 2000 server.

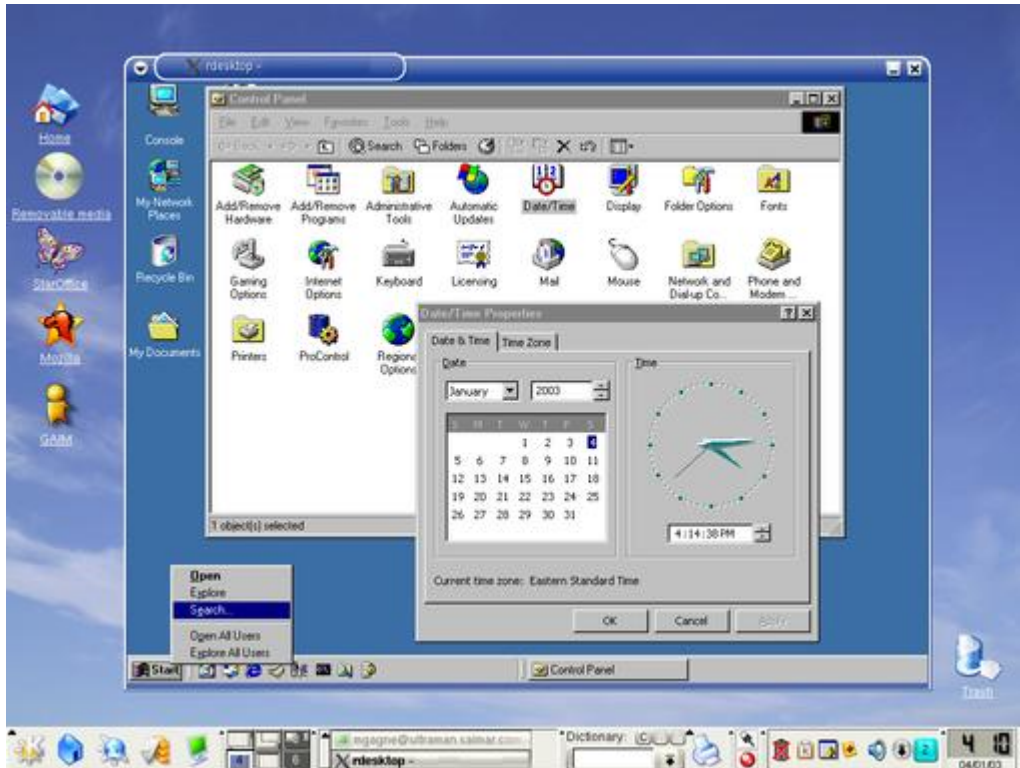


Figure 5. Windows Terminal Server Access Using rdesktop

As you can see, *mes amis*, with a network connection and your Linux system, you are never far away. It is just like being there.

Mon Dieu, has the time passed so quickly. As I cannot connect to your systems and pour you a glass of wine, I must attend to it here before François and I close the restaurant for the night. François, would you be so kind as to refill our guests' glasses a final time? Until next time, *mes amis*, let us all drink to one another's health. *A votre santé! Bon appétit!*

Resources

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

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

[Advanced search](#)

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

[Advanced search](#)

rsync, Part II

Mick Bauer

Issue #108, April 2003

Setting up rsync modules at the filesystem level and making connections.

Last month we covered setting up an rsync server for anonymous access. Listing 1 shows the sample rsyncd.conf file from last month, illustrating some options particularly useful for tightening security. Returning to our example, here's a word about setting up rsync modules (directories) at the filesystem level. The guidelines for doing this are the same as those for anonymous FTP chroot environments. The only exception is that no system binaries or configuration files need to be copied inside them for chroot purposes, as is the case with some FTP servers.

[Listing 1. Sample rsyncd.conf File](#)

The rsync configuration file needs only a little customization of paths and allowed hosts to start serving files to anonymous users. But that's a pretty narrow offering. How about accepting anonymous uploads and adding a module for authenticated users? Listing 2 outlines how to do both.

[Listing 2. Additional rsyncd.conf Modules](#)

First, we have a module called incoming, whose path is /home/incoming. The guidelines for publicly writable directories (see "Tips for Securing Anonymous FTP" in *Building Secure Servers with Linux*) apply, but with one important difference: for anonymous rsync, this directory must be world-executable as well as world-writable, that is, mode 0733. If it isn't set this way, file uploads fail without any error being returned to the client or logged on the server.

Some tips that apply for configuring FTP are to watch this directory closely for abuse and never make it or its contents world-readable. Also, move uploaded files out of it and into a nonworld-accessible part of the filesystem as soon as possible, perhaps with a cron job.

The only new option in the [incoming] block is transfer logging. This causes rsync to log more verbosely when actual file transfers are attempted. By default, this option has a value of no. In addition, the familiar option read-only has been set to no, overriding its global setting of yes. No similar option exists for telling rsync this directory is writable; this is determined by the directory's actual permissions.

The second part of the example defines a restricted-access module named Audiofreakz. There are three new options to discuss here. The first option, list, determines whether this module should be listed when remote users request a list of the server's available modules. Its default value is yes.

The other two new options, auth users and secrets file, define how prospective clients should be authenticated. rsync's authentication mechanism, available only when run in daemon mode, is based on a reasonably strong 128-bit MD5 challenge-response scheme. This is superior to standard FTP authentication for two reasons. First, passwords are not transmitted over the network and therefore are not subject to eavesdropping attacks. Brute-force hash-generation attacks against the server are theoretically feasible, however.

Second, rsync doesn't use the system's user credentials; it has its own file of user name-password combinations. This file is used only by rsync and is not linked or related in any way to /etc/passwd or /etc/shadow. Thus, even if an rsync login session is somehow compromised, no user's system account is directly threatened or compromised unless you've made some poor choices regarding which directories to make available using rsync or when setting those directories' permissions.

Like FTP, however, data transfers themselves are unencrypted. At best, rsync authentication validates the identities of users, but it does not ensure data integrity or privacy against eavesdroppers. To achieve those goals you must run it over either SSH or Stunnel.

The secrets file option specifies the path and name of the file containing rsync user name-password combinations. By convention, /etc/rsyncd.secrets commonly is used, but the file may have practically any name or location—it needn't end, for example, with the suffix .secrets. This option also has no default value; if you wish to use auth users, you also must define secrets file. This example shows the contents of a sample secrets file:

```
watt : shyneePAT3  
bell : d1ngplunkB00M!
```

Contents of a Sample `/etc/rsyncd.secrets` File

The `auth users` option in Listing 2 defines which users, among those listed in the secrets file, may have access to the module. All clients who attempt to connect to this module, assuming they pass any applicable `hosts allow` and `hosts deny` ACLs, are prompted for a user name and password. Remember to set the permissions of the applicable files and directories carefully, because these ultimately determine what authorized users may do once they've connected. If `auth users` is not set, users are not required to authenticate, and the module is available over anonymous `rsync`. This is `rsync`'s default behavior in `dæmon` mode.

And that is most of what you need to know to set up both anonymous and authenticated `rsync` services. See the `rsync(8)` and `rsyncd.conf(5)` man pages for full lists of command-line and configuration-file options, including a couple I haven't covered here that can be used to customize log messages.

Using `rsync` to Connect to an `rsync` Server

Lest I forget, I haven't explained how to connect to an `rsync` server as a client. This is a simple matter of syntax; when specifying the remote host, use a double colon rather than a single colon and use a path relative to the desired module, not an absolute path.

For instance, to revisit the scenario in last month's example, in which the client system is called `near` and the remote system is called `far`, suppose you wish to retrieve the file `newstuff.tgz` and `far` is running `rsync` in `dæmon` mode. Suppose further that you can't remember the name of the module on `far` in which new files are stored. First, you can query `far` for a list of its available modules, as shown below:

```
[root@near darthelm ]# rsync far::  
public          Nobody home but us tarballs  
incoming       You can put, but you can't take
```

(Not coincidentally, these are the same modules we set up in this month's examples; as I predicted in the previous section, the module `Audiofreakz` is omitted.) The directory you need is named `public`. Assuming you're right, the command to copy `newstuff.tgz` to your current working directory would look like this:

```
[yodeldiva@near ~]# rsync far::public/newstuff.tgz .
```

Both the double colon and the path format differ from `SSH` mode. Whereas `SSH` expects an absolute path after the colon, the `rsync dæmon` expects a module name, which acts as the “root” of the file's path. To illustrate, let's look at the same command using `SSH` mode:

```
[yodeldiva@near ~]# rsync -e ssh \
far:/home/public_rsync/newstuff.tgz .
```

These two aren't exactly equivalent, of course; whereas the rsync daemon process on far is configured to serve files in this directory to anonymous users (i.e., without authentication), SSH always requires authentication (although this can be automated using null-passphrase RSA or DSA keys, described in Chapter 4 of *Building Secure Servers with Linux*). But it does show the difference between how paths are handled.

Tunneling rsync with Stunnel

The last rsync usage I'll mention is the combination of rsync, running in daemon mode, with Stunnel. Stunnel is a general-purpose TLS or SSL wrapper that can be used to encapsulate any simple TCP transaction in an encrypted and optionally X.509-certificate-authenticated session. Although rsync gains encryption when you run it in SSH mode, it loses its daemon features, most notably anonymous rsync. Using Stunnel gives you encryption as good as SSH's, while still supporting anonymous transactions.

What About Recursion?

Stunnel is covered in-depth in Chapter 5 of *Building Secure Servers with Linux*, using rsync in most examples. Suffice it to say that this method involves the following steps on the server side:

1. Configure rsyncd.conf as you normally would.
2. Invoke rsync with the --port option, specifying some port other than 873 (e.g., **rsync --daemon --port=8730**).
3. Set up a Stunnel listener on TCP port 873 to forward all incoming connections on TCP 873 to the local TCP port specified in the previous step.
4. If you don't want anybody to connect "in the clear", configure hosts.allow to block nonlocal connections to the port specified in Step 2. In addition, or instead, you can configure iptables to do the same thing.

On the client side, the procedure is as follows:

1. As root, set up a Stunnel listener on TCP port 873 (assuming you don't have an rsync server on the local system already using it), which forwards all incoming connections on TCP 873 to TCP port 873 on the remote server.
2. When you wish to connect to the remote server, specify localhost as the remote server's name. The local Stunnel process now opens a connection to the server and forwards your rsync packets to the remote Stunnel

process. The remote Stunnel process decrypts your rsync packets and delivers them to the remote rsync daemon. Reply packets, naturally, are sent back through the same encrypted connection.

As you can see, rsync itself isn't configured much differently in this scenario than anonymous rsync would be—most of the work is in setting up Stunnel forwarders.

Resources

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

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

[Advanced search](#)

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

[Advanced search](#)

Subcontinental Smackdown

Doc Searls

Issue #108, April 2003

In the global fight between Tux and Bux, where's the best place to bet on the penguin? Try India.

Linux for Suits

Subcontinental Smackdown

On November 11, 2002, the William and Melinda Gates Foundation announced a commitment of \$100 million towards HIV/AIDS prevention in India. The announcement came on the eve of a highly publicized visit to India by William Gates himself. Timed for release with his arrival there, Microsoft announced it would be "deepening its India commitment by making investments of approximately \$400 million US over the next three years in several aspects of its India business including education, partnerships, innovation, localization and the Development Center." So, did it work? The better question might be: how could it work?

Here in the US, spending on information technology (IT) is notoriously down. Although there's a great deal of whining about this by CEOs and IT technology suppliers, the conditions hardly compare with those in India, where the \$600 US price of a cheap PC might amount to a year's salary.

No doubt the cheap, black-market pricing of Windows' earlier generations did much to allow Microsoft to spread as far as it has in India. But the *New York Times* reports the distance isn't all that great, because there still are only about four million PCs in a country with a population exceeding a billion (and expected to pass China by the middle of the century).

But unlike earlier versions of the OS, Windows XP is built not to work unless the user calls or logs in to have the software turned on by Microsoft's authentication process. If Microsoft has its way, all copies in India will be paid for.

Windows XP isn't cheap. And even though it's competing with a free OS, Gates said Microsoft had no plans to discount the OS beyond the usual breaks for educational organizations. Given the competitive situation, how can Microsoft win against Linux in an economy where MS goods carry luxury pricing?

I don't know, and I don't much care—that's Microsoft's problem. What's interesting to me is the other stuff that borrowed interest in Microsoft's problem brings up, such as the potentially leading role that Indian developers are likely to play in the Linux movement.

There are some very ambitious Linux developers in India. Take Rajesh Jain, for example. After I wrote about India on the *Linux Journal* web site, a reader wrote to clue me in about what Jain is up to with his new company, Emergic:

Like Michael Robertson, he was in the internet news business with IndiaWorld, which he subsequently sold. I think that along with Lindows he is one of the folks trying out something in a different fashion with Linux, and there is definitely a huge market at the lower end if he makes the right (whatever those might be) moves.

Here's how Jain describes his plans:

Emergic is about creating a software platform that brings down costs of technology by a factor of 10, thus making it affordable for consumers and enterprises in the world's emerging markets.

Emergic is about realizing Bill Gates' vision of "a computer on every desktop and in every home"—a vision that has not yet gone beyond the world's 10,000 largest companies and 500 million consumers, most of whom are in the world's developed markets.

Emergic is going to become the computing platform for the next 500 million consumers and the world's 25 million small and medium [-sized] enterprises, who have not been able to adopt technology because of its dollar-denominated pricing.

Emergic is targeted at the world's emerging markets, because they are where technology has not yet penetrated deeply, and yet, for whom, technology offers perhaps the last opportunity to better integrate into the world's value chain and improve the standard of living for their people.

Jain wants to drive the cost of hardware down to \$125-150 US or less and software down to \$5-10 US/month. And he wants to do it by leveraging old computers as thin clients for running applications, all the usual Linux suspects. He explains:

What is different about thin clients this time around? After all, they've been talked about ever since computing began.

The major difference is the re-use of older hardware. We use older cars, older manufacturing plants, older homes, but we don't tend to use older computers.

The world's developed markets have been saturated with technology. New PC sales now imply upgrades, creating a huge supply of older computers. (These PCs) still have a lot of life left in them—after all, they are no more than a few years old [and they] can be available for \$100 US or so in large numbers or \$125-150 US in smaller quantities.

He's counting on broadband and talking mostly about enterprise environments:

Server-based computing using Linux is now possible because LAN speeds have gone up to 100Mbps, enabling the transfer of a lot more data over the same network. The result is that a “thick server” (which is actually a new desktop with 1GB RAM and two hard disks in a software RAID configuration) can easily support 30-40 users. Such a server would cost about \$1,500-2,000 US, implying a per-client cost of no more than \$50 US.

Taken together, the thin client and thick server combination not only brings down the cost of both hardware and software by 90%, but it also provides the IT manager complete control of the client desktop from the server. What every user sees on their thin client can be standardized and controlled from the thick server itself.

There's a lot more to his plans, but rather than focus on them, let's look at some other things that are happening in the culture itself. For example, consider the work being done at the Indian Institute of Management in Ahmedabad. Writes Peter Day in *BBC News*:

The Indian Institute of Management is not merely for potential internet billionaires, and the students in their new blue gowns are not the reason for my journey.

I go to Ahmedabad to have lunch with a tableful of some of the most ingenious people I have ever met—inventors and gadgeteers from the fields and villages of rural India where 700 million of its one billion people still live. Over rice and dhal and vegetables eaten with the hand, they talk excitedly about their inventions and ideas.

They are gathered under the auspices of the Society for Research and Initiatives for Sustainable Technologies and Innovation. The acronym spells Sristi, the Sanskrit word for creation.

Sristi is the creation of Anil Gupta, a professor from the Institute whose Honeybee Network has been collecting creative ideas from around rural India for the past ten years. The idea database now exceeds 10,000.

Then there are Dr Sugara Mitra's hole-in-the-wall experiments (see "Natural Forces" from the March 2002 issue), by which street urchins in New Delhi and elsewhere around the country learn computing from free public kiosks.

My points: 1) We're talking about highly resourceful people here—a lot of them. 2) We're also talking about values very much in line with what Linux, free software and open source are all about. That's why I wouldn't be surprised to see Tux beat Bux in a big way here, first.



Doc Searls is senior editor of *Linux Journal*.

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

[Advanced search](#)

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

[Advanced search](#)

Linux Distributions Agree on Standards

Scott McNeil

Issue #108, April 2003

Community-built software and community-built standards are two sides of the same coin. Standards help ensure that the freedom to invent, the essence of open source and of Linux, doesn't compromise the ability to write software that works together effectively.

Way back in 1997, a group of Linux software developers were pondering what could be done to circumvent the minor but troubling variations between the different Linux distributions. Not only that, but they were also contending with differences between versions of a single Linux distribution.

For the group's free software developers, the issue was finding the time to build new functionality and enhancements rather than spending hours verifying that their software worked on all the Linux distributions.

For the non-free software developers, the issue was the same, but they also had paying customers to placate and employees to take care of. Something had to be done.

Fortunately, almost everyone agreed, from upstream authors to Linux distributions to users. Soon thereafter the Linux Standard Base (LSB) Project was formed, with Alan Cox designing the web site, Bruce Perens taking the leadership role and Jon "maddog" Hall offering guidance. Things seemed fine, with Linus Torvalds behind the effort, but this group of pioneers didn't realize just how huge a project they had signed on to. Not only did they need to create a standard that would meet the needs of developers, distributions, businesses and users, but they had to make it really work, and they only had one chance to do it right.

Fast-forward to the year 2000. The LSB was at version 0.02 and was being approached by a group of developers wanting advice for creating a Linux internationalization standard. After a few discussions it was apparent that a

new format was needed, something that would bring in more resources to both efforts while allowing them to remain independent and community-led. This was the genesis of the Free Standards Group.

The Free Standards Group is a California nonprofit corporation dedicated to accelerating the use and acceptance of open-source technologies through the development, application and promotion of standards.

Soon after its founding in late 2000, the Free Standards Group acted as a galvanizer for free and open-source developers and the IT industry alike. Activities around the development of the LSB and Openi18n, the Open Internationalization Initiative, really began to take off. By the close of 2001, both groups had completed version 1.0 of their standards and were confident they would meet with widespread adoption. This confidence was primarily because the targeted adopters were the same people and companies that built the standards. Developers like Ted Ts'o, Stuart Anderson and Dan Quinlan and companies like Red Hat, SuSE, HP and IBM all put their resources into this effort.

These were not efforts for simply documenting a specification; rather, they were creating a formal comprehensive behavioral description of the Linux system and a method for building on to it and proving it. For example, the LSB includes test suites for the operating system, applications and build environment. It also includes a build environment, sample implementation, application battery and full documentation. Here is a breakdown of the pieces:

- **Written specification:** defines the behavior of an LSB-compliant operating system. It does not say which version of a kernel, library or other core element should be used, only the ways each piece will behave. This allows for developers to have to be concerned only with the APIs and APIs of the operating system.
- **Test suites:** include tests for the operating system, applications and build environment.
- **Build environment:** an isolated environment that developers chroot into to build compliant applications.
- **Sample implementation:** an isolated environment that developers chroot into to test run compliant applications.
- **Application battery:** a collection of open-source applications run to stress test compliant operating systems.

About six months after the release of the complete LSB, LSB Certification was launched. Certification gave vendors of both Linux distributions and Linux-based applications a method for verifying and displaying that their products

adhered to the standard. Within six weeks of launching LSB Certification, Mandrake, Red Hat and SuSE had applied for and passed LSB Certification.

Today, every major Linux distribution vendor has applied for and achieved LSB Certification. The debate about fragmentation among the Linux distributions can now come to a close. Application developers can be assured that when they build to the LSB, their applications will run unmodified on any LSB-Certified system. Users will benefit from compatibility among the distributions and a larger body of applications.

Despite its great success in the adoption of its standards, the Free Standards Group and its LSB and Openi18n Workgroups are not sitting still. We are moving forward in extending our existing standards and taking on new tasks such as printing and desktop standards.

If you have any interest in the future of Linux you can join us. Membership is open to individuals, nonprofits (including educational institutions), companies and government agencies. To find out more, visit www.freestandards.org.

Scott McNeil is one of the founders and executive director of the Free Standards Group.

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

[Advanced search](#)

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

[Advanced search](#)

Kylix 3.0 Enterprise (with C++)

Dragan Stancevic

Issue #108, April 2003

Product review of new Kylix release.

Kylix is a powerful RAD (rapid application development) tool. All three flavors of Kylix 3.0 (Open, Professional and Enterprise) come with both C++ and Delphi (Pascal) compilers. The Enterprise edition I reviewed comes with more than 190 components for rapid application development. On top of generic GUI-building components, it also comes with Borland's dbExpress architecture, which enables native access to DB2, Oracle9i, Informix, Informix SE, InterBase, MySQL and PostgreSQL databases. It also comes with BizSnap, WebSnap and DataSnap components, which enable easy development of web services that interoperate with enterprise databases.

The Kylix 3.0 Enterprise package contains a lot of documentation:

- *Quick Start Guide*: an introduction to the product. You will learn how to customize the IDE, and it will also give you an idea of the terminology used to describe various parts of the user interface and its functions.
- *Kylix Developers Guide*: a decent-sized book with in-depth information on usage and development with Kylix. It contains numerous code examples in both C++ and Delphi syntax. It also gives a rather detailed description of the CLX component library. Most of the CLX components are portable between Windows and Linux. For the components that are not portable, this guide has a whole chapter dedicated to porting applications from Windows to Linux.
- *Delphi Language Guide*: the name of this guide says it all. It comes in handy if you've never used Object Pascal but are interested in learning. It also can be a good reference if you are a Delphi programmer.
- CLX (pronounced "clicks") Object Hierarchy poster: this poster shows in an easy-to-read tree view how all the CLX components (objects) fit together. It uses color coding to represent the editions of Kylix in which the objects

are available. The Enterprise edition has the most components, and the Open edition has the least.

- *Borland Solution Partner Resource Guide*: creating components for Kylix is a business in itself. Many different companies write various components for Kylix. For a few hundred dollars, most of the solution partners will provide you with objects that can drastically cut down on your development time. A full list of solution partners is available at www.BorlandSolutions.com.
- Kylix Enterprise (CD): contains the Kylix development executables, libraries and other resources, including source code for all the components. The full component source is useful, but because all the components were written in Delphi, they might be a bit hard to understand if you are not a Pascal buff. The CD also contains both C++ and Delphi versions of the compiler and a lot of sample code. After installing the software, take a look at the examples in the `kylix3/examples/` directory and the tutorials in the `kylix3/documentation` directory.
- Companion Tools (CD): I definitely recommend browsing through this CD as it includes a lot of nice tools. This software doesn't belong to Borland; the tools were written by individuals, groups or corporations. Here you will find various components such as compression, profiling, scripting and game components. Each and every tool comes with its own license, and it's nice to find a lot of open-source code under the GPL and LGPL.
- Enterprise Server (CD): here you will find Borland's application server. The server comes with a development license; a deployment license key needs to be obtained separately.
- Rave Reports (CD): this visual designer by Nevrona Design lets you create custom reports. Once you start the designer, you can point and click to design the look of your report. You can generate reports through different data sources, including database lookups. Once you have your design ready, you save the design into a file, which is later used by calling on the Rave components to generate a report.

Installation

On the installation CD are several text files that you should probably read before installing. They contain descriptions of caveats associated with the installation, development, building and deployment process. After reading the text files on the CD, I ran the installation script. The software can be installed in text mode or graphical mode under X. The script will check to see if it can make a connection to the X server. If it can, the installation process will run under X. This portion of the installation actually uses the Loki installer, and everything is pretty straightforward. One problem was that it didn't create the KDE icons, despite the fact that I checked the box for the installer to do so.

First Run

When you run the command **startbcb** to bring up the C++ version of the IDE, a registration window will pop up. After filling in the data, the on-line registration went without a problem. After the registration window is closed, a nice splash window pops up. It takes quite a bit of time to load the IDE, so I can see the need for a splash screen. When the development environment is fully loaded, a default project is generated.

The IDE is comprised of several floating windows. The main window is docked to the top of the desktop, and it contains all the menus and tabs with the components that are available for use. The second window is the Object Inspector, which shows all the properties and events of an object. A property of a visible component is, for example, what color it should be, its placement rules, caption and more. Under the Events tab of the Object Inspector, you can see all the events this component supports, such as On Click, On Start Drag, On Drag Over and more. The third window is the code editor that shows files that are a part of your program. There is a tab for each open file plus a tab called Diagram. You can drag and drop components onto the Diagram tab from the Object Tree View to create diagrams of your project. The fourth window is the Object Tree View, which shows the tree hierarchy of your program as you add components to it. For a visual representation, take a look at Figure 1.

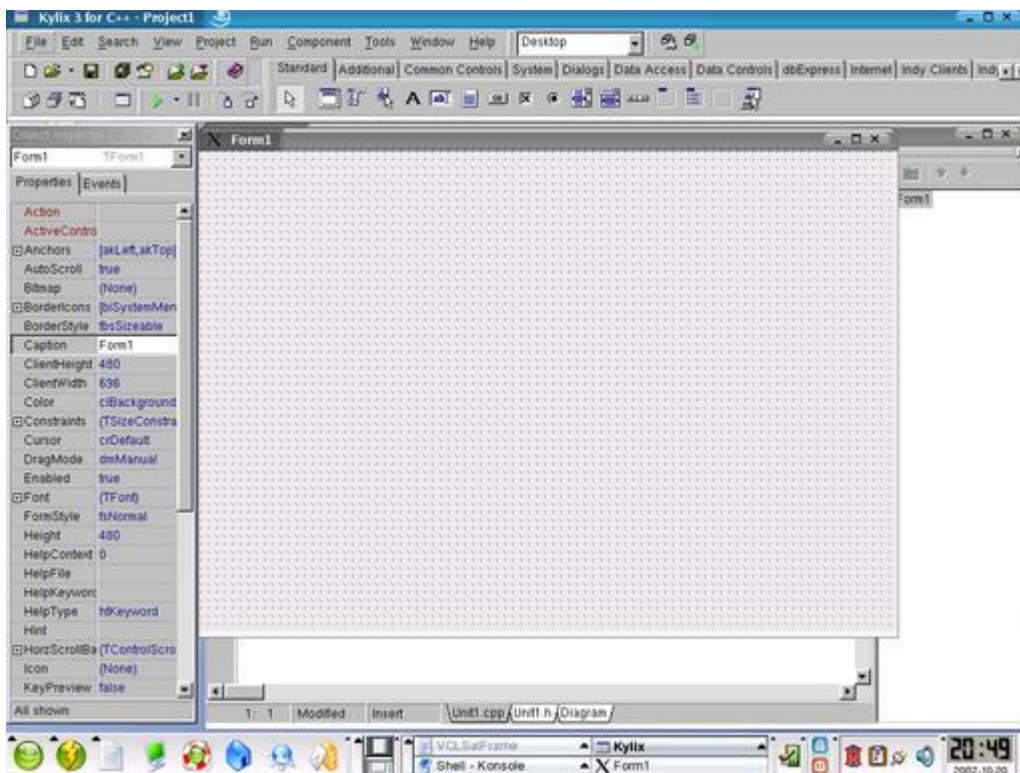


Figure 1. A C++ Application in the IDE

First Compilation

I decided to press the Run button with the default project. It turns out that my default installation of Linux didn't install the glibc-devel package. As you can imagine, this generated a lot of compile errors. It's not a big deal, but I would expect the installer to warn me about missing package dependencies. I installed the RPM and pressed the Run button one more time. This time, the compiler finished, but the linker had problems finding libX11.so. I knew exactly what was wrong; reading all the Readme files before the installation paid off. SuSE installer didn't create the symbolic link libX11.so to libX11.so.6. After I created the link manually, everything worked. I was ready to write, or should I say "point and click", some code.

Building Projects

The whole IDE interface might be a bit confusing at first. After you get a feel for it, you will find it's really easy to build applications. Application design works exactly like a WYSIWG editor. I grab a few components from the component bar and drop them onto my form. I assign one or two of the components properties. What I am looking at is an application that I have not even compiled yet, but the database lookups are running. Simply setting the Active properties to true on the database components was enough to get queries going. I can see how my application would look if I had actually compiled and ran it. Take a look at Figure 2 to see what I mean.

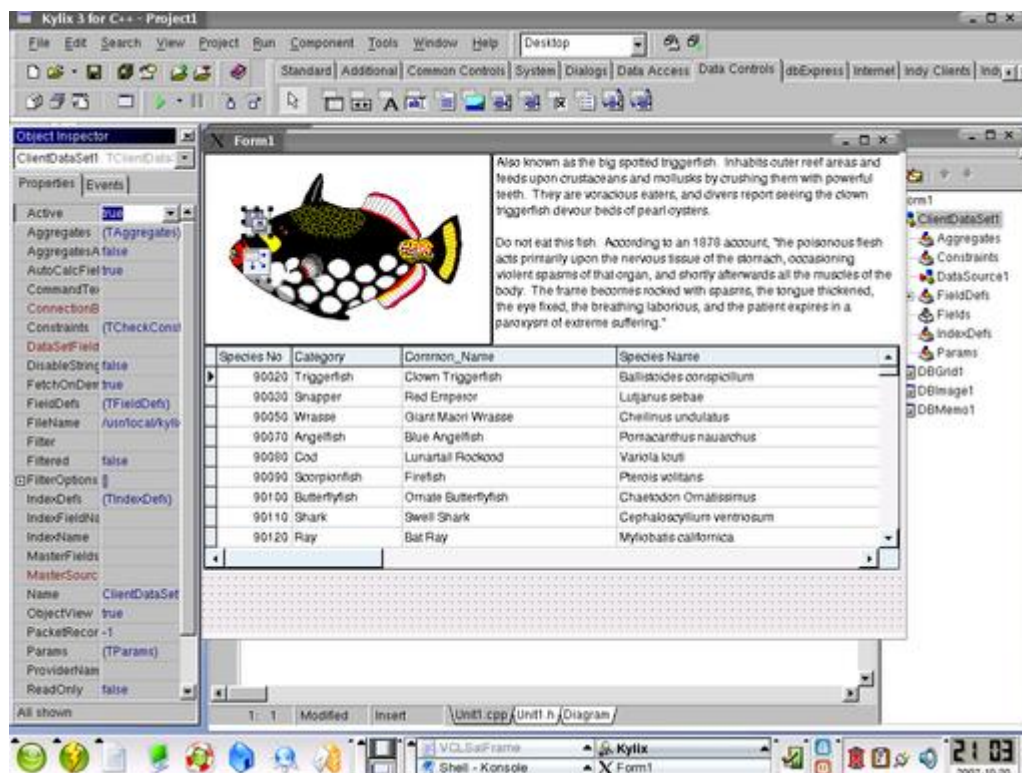


Figure 2. Previewing an Application with Real Data before Compiling

Coding

After clicking around I decided that I wanted to see how the actual coding experience is with the tool. I grabbed a button from the component bar and placed it onto my form (application main window). When I double-clicked this button, the Object Inspector switched to the Events tab and selected the OnClick event. Meanwhile in the code editor, a code framework for the click event was built. My keyboard focus was placed at the function block start, ready for code entry. I wanted the application to close when I clicked the button that I just placed on the form. I started typing "Application->" and a small window popped up in the editor. It listed all the functions and properties of my application instance called "Application" (Figure 3).

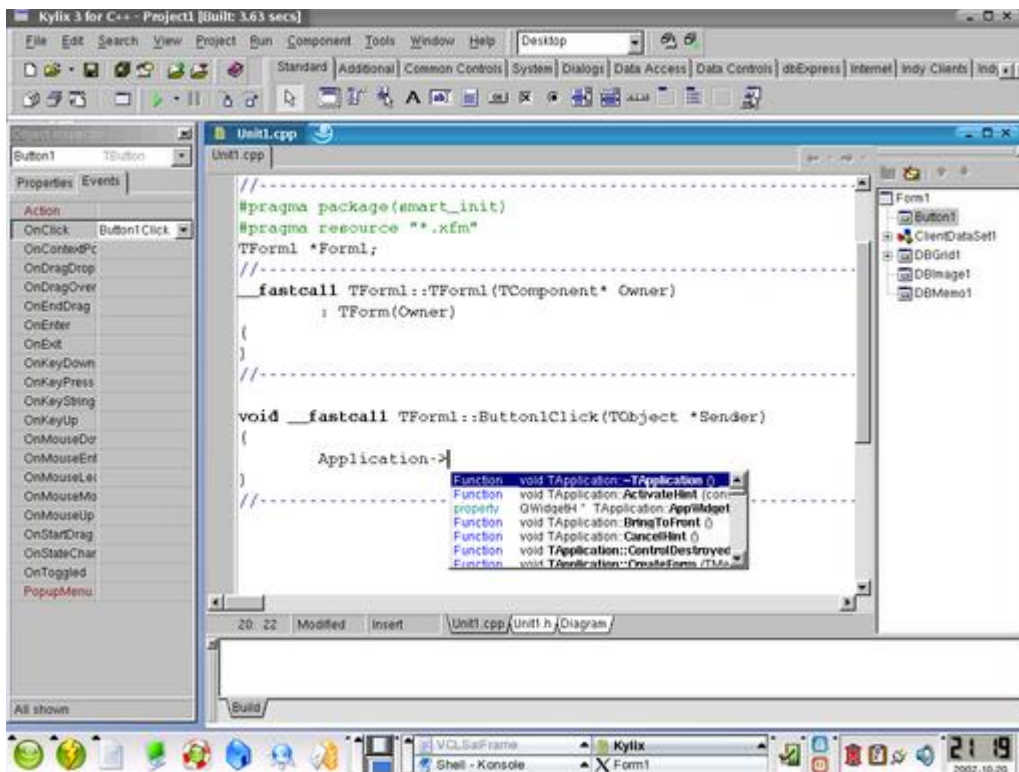


Figure 3. Automatic Function Name Completion in the Editor

As I typed in the letter T of the function name (Terminate) that I wanted to call, the code-completion feature eliminated properties and functions that didn't match up. When I saw what I was looking for, I scrolled up and down through the list and pressed Enter. That filled in the code in the editor, then I need to add the semicolon at the end. I thought to myself "that's nice but how about if I spice it up a bit?" I included a Linux header file called `utsname.h` and instantiated a structure of type `utsname` and name `tst`. I typed in `tst.` and waited to see what would happen. The same window popped up and listed all the structure members and their types (e.g., `char[65]`). This comes in handy in those "what's that function called again?" moments.

Conclusion

Borland introduced Delphi for Windows a long time ago, with many of the Kylix features. Years later they introduced the C++ Builder for Windows, which as the name suggests, was the C++ version of Delphi. It is nice that a C++ edition of Kylix for Linux is finally here. All in all, I like the design of Kylix. Despite the fact that it's not a new concept, it still has benefits.

After playing around with Kylix some more I found a few bugs in the software. When trying to assign images to components, the application would freeze. I am guessing it's possibly some sort of synchronization issue because attaching to the Kylix process with strace brought it back to life. I also found that the code-completion window sometimes refused to pop up. To put things in perspective I must say that Kylix was not certified to run on the distribution that I had freshly installed on my laptop. I used SuSE 8.1, although Kylix 3.0 was certified to run on SuSE 7.3.

As a final thought, if you are looking into evaluating development tools for enterprise applications I would recommend putting Kylix 3.0 on your "tools to evaluate" list.

Product Information

Resources

email: visitor@xalien.org

Dragan Stancevic is a kernel and hardware bring-up engineer in his late twenties. Although Dragan is a software engineer by profession, he has a deep interest in applied physics and has been known to play with extremely high voltages in his free time.

Archive Index Issue Table of Contents

Advanced search

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

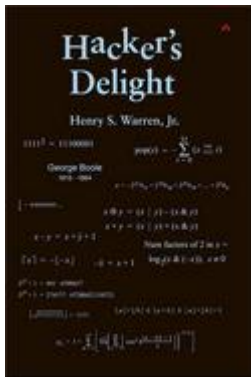
[Advanced search](#)

Hacker's Delight

Michael Baxter

Issue #108, April 2003

The book is a wonderful collection of techniques for any programmer looking to improve efficiency for key algorithms.



Book Review: *Hacker's Delight* by Henry S. Warren, Jr.

Boston, Pearson Education, Inc., 2003

ISBN: 0-201-91465-4

\$39.99 US (hardcover)

Hacker's Delight is a treasure trove for learning how to write efficient code. Over the course of 16 chapters and two appendices, fiendishly clever algorithms are illustrated through numerous examples coded in C and with graphics, along with the mathematical theory that supports the techniques.

The introduction describes an instruction set and execution efficiency model on which the rest of the work is based. This provides a useful means for assessing computational efficiency on most modern computers. Then the stage is set for a chapter about many little issues surrounding bit-level representation. Next, a chapter each is provided for algorithms involving power-of-two boundaries and arithmetic bounds.

Efficient techniques for bit-level manipulations, such as counting the bits in a word, are described in the next three chapters, and the next four chapters form an étude about some key arithmetic operators and functions, including efficient multiplication and two kinds of integer division. A chapter follows covering algorithms for computing key elementary functions on integers. This includes logarithms, exponentials, plus square and cube roots.

In some specialized applications, alternative representations for the meaning of the bits can offer an advantage. Unusual number system bases and Gray codes are described in detail in two chapters. Gray codes, for instance, are useful for enumerating states in finite-state machines where only one bit changes per state transition.

One chapter provides an algorithmic glimpse on a variant of the Peano curve called the Hilbert curve, which is an interesting related space-filling curve. The Hilbert curve is amenable to recursive algorithms and has some bit-level computational advantages for representing spatial distances in coordinates. These algorithms find usefulness in image processing, rendering and compression.

A nice summary of IEEE Std 754-1985 floating-point arithmetic includes a procedure for comparing floating-point numbers using only integer operations, formulas for computing the probability density function for the number of leading digits in select ranges of representable numbers and a handy table of miscellaneous number values represented in hex for both single- and double-precision floating point.

The last chapter provides algorithms for computing prime numbers, using Willan's and Wormell's Formulas. Primes have uses in hashing algorithms and cryptography, among other things.

Two appendices provide 4-bit arithmetic tables and a more detailed description of Newton's Method for function approximation. The arithmetic tables are a handy way to envision the work done in algorithmic steps that are a fraction of a typical word length. The bibliography also is rich, listing many original papers for arithmetic, number theory and the techniques embellished within the book.

You can employ these techniques to attain mastery of some important inner-loop code, while enjoying the beauty of arithmetic algorithms. The author is a veteran of IBM, and his programming tricks are born of experience across four decades, from the IBM 704 through the PowerPC. The book is a wonderful collection of techniques for any programmer looking to improve efficiency for key algorithms in areas such as compiler development, databases, arithmetic for image and signal processing, and code libraries.

—Michael Baxter

email: mab@cruzio.com

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

[Advanced search](#)

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

Advanced search

Letters

Various

Issue #108, April 2003

LJ readers sound off.

Zaurus Fans Unite

Excellent article ["Must-Have Zaurus Hardware and Software", *LJ*, January 2003]! Very informative and a lot of detail. I have just bought my Zaurus, and I downloaded a lot of utilities and was able to set my Zaurus's configuration with the help of Guylhem Aznar's article. Thank you.

—Vinh Duong

Be Careful with ptrace

In "Playing with ptrace, Part II" [*LJ*, December 2002], Pradeep Padala talked about injecting code into a process and finding some "free space" to put it in to. It's worth noting that the space referred to is not really "free"; it's usually either the cleared space used for global storage in the executable and its shared libraries or the C library's heap storage area. In any case, writing over this data and not restoring it before allowing the execution to continue (as may seem reasonable at first) could cause all sorts of weird behaviour, including program crashes.

—Shaun Clowes

FreeS/WAN Updates

Just got my January 2003 issue of *LJ* and was quite surprised to see a FreeS/WAN article included—nice work! I was really happy to see you used the RSASigs in the examples instead of preshared secrets, a welcome change from the usual and insecure examples I've read in the past. I maintain www.freeswan.ca, an alternate source of information, patches and prepatched versions of FreeS/WAN for interoperation with many devices. Freeswan.org

now ships RPMs for Red Hat 7.x and 8.x for all kernel combinations. These include only the ipsec.o modules and user-land tools and don't replace your vmlinuz and grub/lilo configs. Folks should update to 1.99, as there was a serious denial-of-service flaw that is now fixed.

—Ken Bantoft

Mick's reply: Thanks very much for your suggestions. Part II appeared in the February 2003 issue, and I doubt this is the last I'll write on the subject!

Surprising Our Readers... with Quality

As a long-term *LJ* reader (fourth year), I am really surprised about the great January 2003 issue—it covers all the stuff that I am interested in without even knowing about it. The GCJ, Screen and DDD/quicksort articles shed more light into the daily use of our beloved Linux platform. Please keep us informed about developments in compilers, debuggers and other development tools to make us more effective in developing new stuff. Keep up the good work.

—Raphael Arlitt, Germany

make penguins && make party

What LUG meeting or BOF session would not be enhanced by penguin canapés and an igloo cheeseball? We humbly submit pseudo-code for building same and an image as proof that it's working code.



Ingredients:

2 packages cream cheese1 cheeseball1 can large black olives, pitted1 can small black olives, pitted1 carrot1 packages toothpicks with yellow or orange fringe crackers1 tin kippered herring (optional symbolic offering to penguins)

```
#!/bin/bash
while hungry;
do (\
    cut_cream_cheese_into_strips_and_cover_cheeseball;\
    make_igloo_entrance_tunnel_from_cream_cheese_strip;\
    use_toothpick_to_sculpt_snow_block_seams;\
    peel_carrot_with_vegetable_peeler;\
    cut_carrot_into_coin-sized_slices;\
    cut_slender_wedge_from_each_carrot_slice_and_reserve_for_beak;\
    slit_each_large_olive_and_stuff_with_cream_cheese;\
    puncture_each_small_olive_and_insert_carrot_wedge_beak;\
    skewer_small-olive_head_large-olive_torso_and_carrot-slice_feet_with_toothpick;\
    arrange_olive_penguins_about_cheeseball_igloo_on_serving_dish;\
    arrange_crackers_on_serving_dish_or_nearby;\
    serve);
done
```

Herring also can be served to set the scene. Herring are one of two things that make penguins contented. This recipe is a clean-room implementation developed by reverse engineering based on a study of olive penguins and a cheeseball igloo served at a party. We hope the process is not patented. In any case, we assert that the recipe is our own work, and we release it under the terms of the GNU Free Documentation License.

—Michael Callaham [penguins] and Jennifer Gentry [igloo]

Keep *LJ* Free of Microsoft Ads

I have just received the February 2003 issue of *Linux Journal*, was reading through the Letters and came across the one with the guy who wants to run Microsoft ads in *LJ*. If I want Microsoft ads, I will go to a Windows magazine. It is true that MS ads cannot harm us, but they are annoying!

—Mitch Anderson

Linux and Land Reform

Kudos for Jon Hall, the *LJ* magazine and the thought behind the GNU/Linux and other free, open-source software movements (“Back to Brazil”, Letters, February 2003). The heavy reluctance against land reform is the root cause of fundamental socioeconomic problems in many parts of the globe, including Southeast Asia, South Asia, Central and South America and Africa. I believe land reform movements in these countries share the same vein that the Linux/GNU movement has. Our country, Japan, had been coerced into doing a total land reform by the US occupation policy in 1940s. It liberated not only the land but the minds of so many common people, enabling the making of the world's second biggest economy and modern industry.

—Hiroshi Iwatani

Origin of “Wardriving”

In two articles [*LJ*, September 2002], Doc Searls incorrectly claims that the terms “wardriving” and “warchalking” were derived from the movie *War Games*. I believe they actually were derived from the term “war dialing”, the process of sequentially dialing a range of phone numbers in search of a modem connect tone. War dialers were utilities that could be left to run and accumulate interesting numbers to be investigated later. These programs were the equivalent of today's internet port scanners.

—Jonathan Hutchins

War Games came out in 1983, and we don't know of an example of the term “wardialing” before that. Although the other “war” terms are derived from “wardialing”, it's likely that “wardialing” came from the movie. Wardialing is still a threat according to a 1998 survey by Peter Shipley: www.dis.org/filez/war.pdf.

—Editor

maddog Inspires One Reader...

Just to say what an inspirational answer to the unfair criticism maddog received from that reader in Brazil [*Letters, LJ*, February 2003]. I really liked maddog's reply, especially the paragraph that starts “I believe...”. I just might put that paragraph on an A4 page above my desk! Great stuff!

—Paul

PS: Not sure what you guys have done, but *LJ* now arrives in my office in-tray the day after you announce it on the web site. It used to take weeks (or longer). Whatever you've done, keep doing it.

...and Loses Another

I don't subscribe to *LJ* to read about politics, especially apologies for terrorists and other assorted anti-American, third-world riffraff. My subscription expires in March 2004, and I won't be renewing.

—Chas

Where Is the Fun?

I see Linux as losing its sense of humor. I am willing to bet a cup of coffee that this is due to the infusion of money from big business. Don't we have a wacky penguin? Don't we have XBill? Isn't that funny stuff? The answer is a qualified yes. It's funny, but it's funny in the same regard that recess is fun—that being because it is at a set time and place and supervised. But Linux never had a recess time before. You could play all day and still get everything done. It's time to get off our collective asses and put the fun back in Linux.

—Mutant

You can't spell FUN without U.

—Editor

What Happened to the Thick LJs?

Before I decide not to resubscribe, is *Linux Journal* going to return to the size that it used to be? I feel that the quality and content of the magazine has dropped in the last year and find it hard to sign up again for another year. The cost has not changed, but the content has!

—Tony

The number of pages of content depends on the number of pages of advertising. More ads fund more content. In a down economy when ads are few, page counts go down everywhere.

—Editor

maddog 3, Brazilian Land Barons 1

Man, when I got to the "I believe in...." paragraph I nearly choked up. I'm glad you took the space to allow his response [*LJ*, February 2003]. Oh, and another good reason for showing MS ads—they're always a good laugh. Good mag.

—Daniel

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

[Advanced search](#)

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

Advanced search

UpFront

Various

Issue #108, April 2003



Gtk-Perl Flash Cards: www.masswire.com/flash.tgz

Flash cards have always been a good way to go through questions. You get the question, provide a response and see if it's correct. That's the theory behind this little application. You can classify the question as difficult or easy, put as many as you'd like in the data file and quiz yourself. The application provides a way to keep track of the number of correct and incorrect responses, total number of questions, a button to randomize the next question and more. Requires: Perl, Perl module Gtk.

—David A. Bandel

Checkbot: degraaff.org/checkbot

This is a simple, lightweight web checker. If you run it with the `--match` option and match your domain name, it will not wander off on all the sites you might be linked to. You can have it mail the results or simply watch it traverse your site picking up the URLs. No more broken links, no matter how complex the site—at least none you don't know about. Requires: Perl, Perl modules `File::Basename` and `LWP`.

—David A. Bandel

Dbmail: www.dbmail.org

If you're running a large mail server with thousands of users, Dbmail may be helpful. Users and mail are all stored in an SQL database; system users do not need to be created. The Dbmail program comes not only with `dbmail-smtp`, a receive-only SMTP daemon (you'll still need `sendmail` or another MTA for outgoing mail), but also with `dbmail-pop3d`, a pop3 server, and `dbmail-imapd`, an IMAP daemon. Instructions for setup are sketchy, so you'll need to figure it out for yourself. Fortunately, that's not too hard to do. Configuration includes things like POP or IMAP before SMTP. Requires: PostgreSQL or MySQL, `libssl`, `libcrypto`, `glibc`.

—David A. Bandel

diff -u: What's New in Kernel Development

In spite of the October 31, 2002 feature-freeze, developers continue to hack on their favorite projects. December saw a number of such developments, some weirder than others. In the POSIX realm, **Krzysztof Benedyczak** and others did some work on implementing **POSIX message queues** to allow processes to communicate more directly with each other. Linux always has maintained a love/hate relationship with POSIX (and other official standards), sticking to the principle that bad ideas should be avoided whether they have an official seal or not. Message queues have not been particularly controversial in the Linux arena, but the UNIX world at large has not always agreed on the proper public interfaces for them. So whatever the ultimate Linux message-queue implementation, there will be permanent issues surrounding attempts to port any applications that use the feature.

Drivers for new hardware are churned out constantly, development series or no. December saw several new drivers for **Via cards** (the 8633 AGP and 8233 onboard sound card) from **Nathaniel Russell** and a framebuffer driver for the **Intel 810 and 815** graphics chips from Antonino Daplas. Overall the framebuffer code did not fare spectacularly well in December, though many patches and

advancements were made. Part of the problem is the basic framebuffer design makes assumptions that simply are not true for certain hardware, and the design issues are hard to correct because a lot of user-space code relies on the existing implementation. But **James Simmons** has been quite active in addressing the issues that can be addressed, and a lot of work by him and others will be in the 2.6 framebuffer code, including some fancy new APIs.

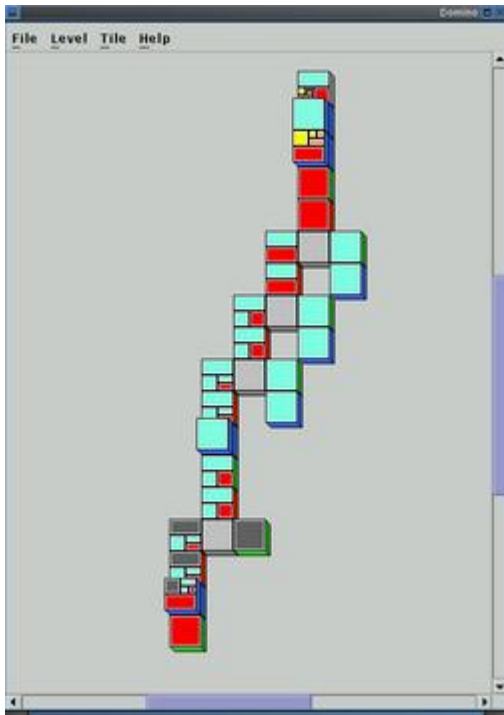
A whole new architecture saw the light of day in December. After a month's intense labor, **Andrey Panin** ported Linux to the **SGI Visual Workstation**. Some might say, as **Alan Cox** did when Andrey announced his work, that such an effort belonged truly in the land of dementia. After all, the VISWS was apparently a flash in the pan, appearing briefly a few years ago and then dropping off the map. But Alan still applied the patch.

Intel's sysenter and **sysexit** instructions, introduced way back with the Pentium II, finally are starting to find support under Linux. Theoretically, they provide a quick way to perform system calls, but in practice it proves difficult to find an implementation that doesn't sacrifice too much of the speed the instructions are intended to save. A lot of progress was made in December, but this is all quite invasive work, and as Alan Cox has said, **Linus Torvalds** appears to be "doing the slow slide into a second round of development work again", as was the case with all other development series.

Speaking of invasive work, it looks as though **Andre Hedrick's** new **IDE subsystem** will be dropped *en masse* into the 2.4 kernel. Normally such an invasive change would be attempted only during a development cycle, but apparently the old IDE code is too nightmarish to live. News of the new IDE code's imminent acceptance into 2.4 was greeted with shouts of jubilation from all sectors. Quite a different reaction from Linus' decision to drop a new virtual memory subsystem into an earlier 2.4 kernel.

In the final months of 2002, some folks decided to set up a **Bugzilla database** to help bring 2.5 to a successful, stable conclusion as soon as possible. Not all developers feel that Bugzilla is the best tool for the job, however. Since the bug database was first set up, it has proven difficult to use in certain ways. Bugs go unclaimed, and developers have trouble finding references to bugs in their areas of interest. In light of this, **John Bradford** decided to start from scratch and implement an entirely new bug-tracking system, designed specifically for the Linux kernel. He chose to focus on automating much of the search facility and enhancing the organization and presentation of bugs to streamline the ability to find bug reports in any particular area of interest. That said, the existing Bugzilla database has its adherents, and as of the end of December 2002, John still had not put together a fully usable replacement.

—Zack Brown



Domino on Acid: www.winterdrache.de/freeware/domino/index.html

This is an extremely difficult game but includes its own tips, hints and tricks manual. You can download this game or play it right on the Internet. It can be invoked as a jar file for a standalone game or called from any Java-enabled browser. If you're like me, you'll have tiles all over the place and be no closer to a solution than when you started. Requires: Java.

—David A. Bandel

LJ Index—April 2003

1. Sum offered (and paid out in January 2003) by Lindows founder Michael Robertson for the successful port of Linux to Microsoft's Xbox by the end of 2002: \$100,000
2. Sum offered by Robertson for a port to Xbox with no hardware modification by the end of 2003: \$100,000
3. Year by which Linux is expected to become the majority server operating system: 2009
4. Position Linux is expected to occupy soon among desktop operating systems: 2
5. Price in rupees of Hewlett-Packard's AMD 1.5GHz Athlon-based Presario home computer, sold with Red Hat Linux: 30,990 (\$645 US, as of January 3, 2003)

6. Price in rupees of Hewlett-Packard's Intel 1.6GHz Intel-based Presario home computer, sold with Windows XP: 40,000 (\$833 US, as of January 3, 2003)
7. Additional discounts on the Linux Presario, in rupees, from "assemblers": 2,000
8. Millions of Google results from a search for "Linux" on November 29, 2002: 41
9. Millions of Google results from a search for "Linux" on January 2, 2003: 59
10. Growth in "Linux" results per day over the same period: 529,412
11. Position of Linux among Google's top ten technology searches in 2002: 4
12. Position of Microsoft among Google's top ten technology searches in 2002: 9
13. Number of applications shown on stage by Sun Microsystems at Comdex Fall 2002: 2
14. Number of applications shown on stage by Sun Microsystems at Comdex Fall 2002 that ran on Linux: 2
15. Percentage of servers on which Linux is expected to run by 2006-2007: 45

Sources

1. 1, 2: Xbox Linux Project (xbox-linux.sourceforge.net)
2. 3: Butler Group Server Operating Systems Report (www.butlergroup.com)
3. 4: IDC (via ZDNet)
4. 5-7: *Financial Express*
5. 8-10: Google
6. 11, 12: Search Engine Watch
7. 13, 14: Doc Searls, reporting from Comdex
8. 15: Meta Group, quoted in *BusinessWeek*

They Said It

When a need comes up for a new file or print server, don't talk about installing a Linux box. Talk about installing a new file or print server. As long as what you implement does the job and works reliably, no one will care how it's done as long as it works.

—Craig Sanders, Debian developer and professional system administrator

A for-profit software company cannot compete with the economics of open source—free is as cheap as it gets. Nor, it turns out, can it compete with open source's quality testing process. Though the pace of open-source development can be languid and tends to create products less functionally rich than their

proprietary counterparts, the stuff gets tested so often and so brutally by so many different people that most open-source programs are judged to be more stable and reliable. In a commodity market, low cost and reliability count more than bells and whistles.

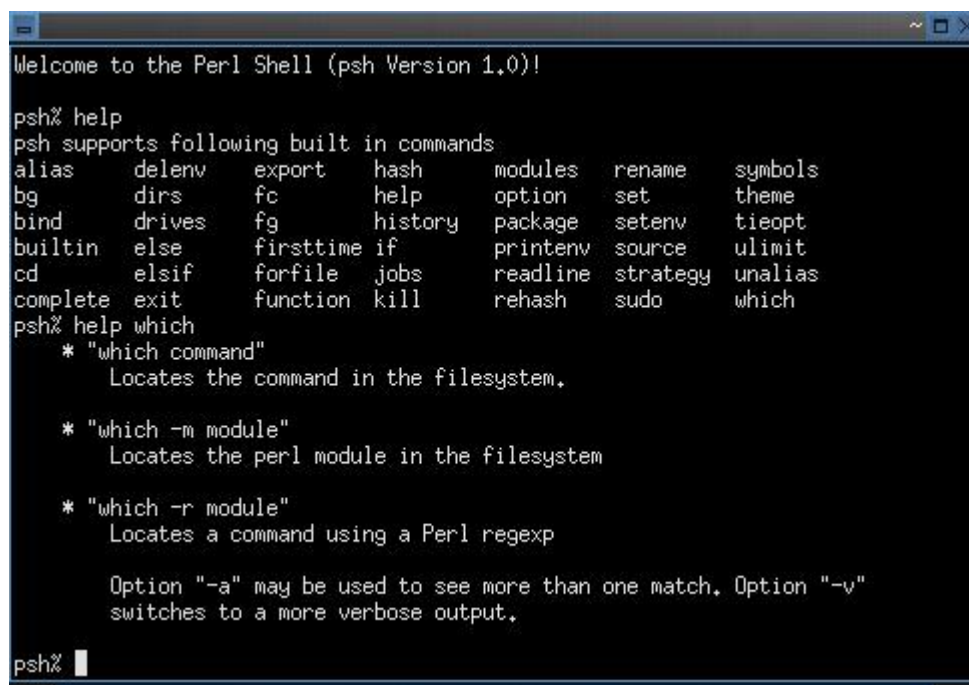
—Christopher Koch, *CIO Magazine*

What I've found is that a Linux administrator who knows what he's doing should be able to administer two to three times the amount of boxes a Windows administrator should be able to administer.

—Brian Schenkenfelder, n+1

Mongolian makes the impossible possible and enters the list at a whopping 15th place (supported). Sanlig Badral, Ochirbat Batzaya, Tegshbayar, Bayarsaihan and the other guys in the Mongolian team have certainly made an impressive start by jumping right in the top crowd with over 95% translated messages!

—Christian Rose, on the GNOME-I18n Mailing List (GNOME is now 100% translated to Mongolian)



```
Welcome to the Perl Shell (psh Version 1.0!)

psh% help
psh supports following built in commands
alias      delenv    export    hash      modules  rename    symbols
bg         dirs     fc        help      option   set       theme
bind      drives   fg        history   package  setenv    tieopt
builtin   else     firsttime if         printenv source    ulimit
cd        elsif    forfile   jobs      readline strategy  unalias
complete exit     function  kill      rehash   sudo      which

psh% help which
* "which command"
  Locates the command in the filesystem.

* "which -m module"
  Locates the perl module in the filesystem

* "which -r module"
  Locates a command using a Perl regexp

Option "-a" may be used to see more than one match. Option "-v"
switches to a more verbose output.

psh% █
```

psh: www.focusresearch.com/gregor/psh

For all the Perl mongers out there, this shell might be for you. It has a number of the features of Bash, but it's a little more Perl-friendly. I've recently started using it as my login shell, so we shall see. It's certainly lighter. If you're skeptical, ask me if I'm using it a year from now. I see a day when Perl may replace all the

other system utilities (if one is so inclined), but until then, I'll be satisfied with a Perl shell. Requires: Perl, BSD::Resources (optional).

—David A. Bandel

```
HW-ADDR: 00:60:1d:f1:40:9e -----> 00:50:ba:d8:22:32
IP-ADDR: 192.168.0.2 -----> 192.168.0.1
IP-Ver4 || Head:0x14 (bytes) || Service(TOS):16 || Length over all:1500
Fragmentation: ID:0x6aa6 - Flags: 0 1 0 - Offset:00000
TTL:064 || Protokoll:006 (TCP) || HeaderCRC:0x4912
TCP-HEADER:
Ports: 0022->50853 (unknown) Seq./Ack. Nr.:0x09e506df / 0x091b4c77
Data-Offset:0x08 Reserved-6Bit:00 Flags:urg-ACK-psh-rst-syn-fin-
Window:0x0000 URG:0x0 Urgent-Pointer:0x0000
Bytes until here: 0x0036 Bytes over all: 0x05ea

01 01 08 0a 00 04 f8 c8 03 9a b3 1e d6 b7 68 c2 4f 54 4d a9 0e 90 8c 7c 24 96
a5 51 ec a7 a6 15 ee f8 e5 2b 19 ee f0 0d 8a f1 e4 e7 bd cd 48 4d 72 f7 f9 1d
87 19 bb c5 9c c1 36 fb ef db d1 8a 5d 3a ad 5e e5 fe a1 ae d1 9c 33 65 63 60
96 c7 c6 35 50 28 17 34 4b a8 34 bb f3 6c 5e 33 84 10 78 bd 26 c2 9d 96 ee 41
a9 64 cb 7e 78 1a ca 26 43 71 db ec 9e 6d 2b b9 6c ba 6d 08 51 27 19 19 3e 56
6a 35 93 25 ca c7 20 53 74 fb 28 47 2f a1 a2 5b 01 72 7e 45 c7 59 bd 64 3e 69
3a c6 cf 42 7d 6a e8 32 5b 6a 7d e3 d3 a4 3d 2d a6 54 82 0e 64 f6 14 c4 04 88
cb 9f e0 03 da 20 e5 fa 42 51 9c aa 6e 72 06 39 31 ce b0 92 b0 43 b7 ad 9a 4c
92 e6 89 39 ec f0 a9 94 18 93 e7 fc d4 77 b0 eb b0 97 9d a7 1f 0f 4c 91 14 97
69 f3 1c b1 db 94 d8 8e 55 05 1e 99 58 10 d6 7a 36 eb 3f 7c 48 37 e2 83 e8 e7
2c e9 ac a0 8c 5f 44 d2 36 d7 63 a2 71 16 fa 6b a2 84 1b 52 da 9c e2 56 20 f6
ac 2d ef 90 7c 6f a4 0b f9 b0 ec 40 e6 a6 80 58 a0 89 2b fb 08 3c 56 06 76 de
84 68 a6 5e ae 25 36 c1 8e 5e 44 19 84 68 a6 5e ae 25 36 c1 8e 5e 44 19 5f 35
48 2c 23 b6 93 53 0b e9 4f fe e4 c2

APS: terminating after 266 packets. (2)

Package summary:
--> IP :000266 ## TCP-IP:000266 UDP-IP:000000 ICMP-IP:000000
--> RSLV:000000 ## RRP-REQ:000000 RRRP-REQ:000000
## RRP-REP:000000 RRRP-REP:000000
--> ICMP:000000 ## ECHO-REQ:000000 TIME-REQ:000000 INFO-REQ:000000
## ECHO-REP:000000 TIME-REP:000000 INFO-REP:000000
## ADDR-MASK-REQ:000000 ADDR-MASK-REP:000000
unreachable ## HOST:000000 NETU:000000 PROT:000000 PORT:000000
unreachable ## FRAGM-NEEDED:000000 SRC-ROUTE:000000 UNKNOWN:000000
redirect ## TO-NET :000000 TO-HOST :000000
redirect ## TOS-NET:000000 TOS-HOST:000000 REDIR-UNKNOWN:000000
other ## EXCEEDED-TIME-LIMIT:000000 PARAMETER-PROBLEM:000000
## OVERALL-ICMP-UNKNOWN:000000
--> OTHER:000000 ## LDDP:000000 SHB:000000 UNKNOWN-FRAMES:000000

Detached SHMEM-Segment (ID=524292)
chiriqui:/home/david/aps-0.19/src#
```

Advanced Packet Sniffer: www.swrtec.de/clinux

As I noted when I originally reviewed this application three years ago, this particular sniffer is unlike tcpdump. Here, you can see the packet payload, which may make a lot of sense to you or none at all (particularly if someone is using an encrypted connection). One of the reasons I most like this sniffer is you can show someone what information is floating around on their network for anyone to read. Requires: glibc.

—David A. Bandel

Mongolian makes the impossible possible and enters the list at a whopping 15th place (supported). Sanlig Badral, Ochirbat Batzaya, Tegshbayar, Bayarsaihan and the other guys in the Mongolian team have certainly made an impressive start by jumping right in in the top crowd with over 95% translated messages!

—Christian Rose, on the gnome-i18n mailing list (GNOME is now 100% translated to Mongolian)

email: david@pananix.com

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

[Advanced search](#)

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

[Advanced search](#)

Tools for Desktop Success

Don Marti

Issue #108, April 2003

Looking at applications that meet the needs of business and standards that are bridging the gap between competing Linux desktops.

The last thing we're going to try to do this month is answer the question, "Is Linux ready for the Desktop?", because only you can answer that. Nobody's going to blow a whistle and make it practical for everyone at once to install desktop Linux.

I've been using Linux on the desktop since around the time Netscape Navigator came out. Unfortunately, many of the applications people use to get their jobs done aren't available on Linux yet. So if you have a lot of data buried in some proprietary format, you might have to keep some proprietary desktops around. In this issue though we're going to provide as much information as possible to help put your company or organization on the path of freedom and self-determination.

Leading by example is Gary Maxwell, who's no Linux or UNIX guru—just a small-business owner looking for stability in his working environment. He's now running his whole commercial writing firm on free software. Find out how he's doing on page 48.

If you don't like the fact that applications written with different toolkits often don't work well together, now there's something you can do about it. Check out our cover and read Marco Fioretti's "The Grand Unified Desktop" article on page 38. Marco received a lot of comments on our web site when he praised Red Hat's Bluecurve desktop for mixing the best of GNOME and KDE, and now he's taking an in-depth look at standards for things like drag-and-drop and configuration files. Don't take sides in the desktop war—follow standards so you can use the applications you like.

Page 44's article comes from the "stuff the editor wanted to learn" pile. Chris Schoeneman has invented what you might call a software KVM switch. It's Synergy, a program that lets you move the pointer to the edge of one system's display and start working on another system. Set your laptop down next to your desktop system, and automatically get more work space without switching keyboards.

Setting up Linux for desktop use still has some tricky parts, and scanning certainly qualifies. On page 54, Michael J. Hammel goes through the intricate dance of setting up a scanner. If you can do this, buy yourself a beverage and consider yourself ready for most Linux tasks.

Whether you're planning to develop software for desktop Linux, run The GIMP or design a web content management system, there's plenty of other good stuff in this issue too.

Finally, you might not think of "Hacking Red Hat Kickstart" (page 83) as a desktop Linux article. After all, the whole point Brett Schwarz makes is you can install and configure a new system without touching the mouse and keyboard even once. But the promise of user control over time-consuming tasks is one important reason people think a Linux desktop is worth the effort in the first place.

Don Marti is editor in chief of *Linux Journal*. Since reading the Bayesian spam-filtering article from last issue, he is most likely to read mail with the words "wrote", "discussion", "mutt", "hardware" or "reform".

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

[Advanced search](#)

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

[Advanced search](#)

Eclecticism for the Masses

Heather Mead

Issue #108, April 2003

Variety being the spice of life and all that, the *LJ* site offers a little something for everyone.

Being eclectic is a good thing, even though the word often is said in a tone of voice usually reserved for words like “lice” and “spellcheck”. The *Linux Journal* web site, however, prides itself on being a compendium of topics related to open-source, free software and, of course, Linux. In any given week, a visitor might find articles ranging from hardware reviews to lessons in spam filtering to how-tos for building one's own VPN gateway. It may sound obvious, but a lot of people are using Linux to do a lot of different things. Our site attempts to provide articles that explain how to do what you're dying to try, as well as introduce topics and projects with which you might not be familiar. We might even be able to help you win an argument.

Say, for instance, someone is giving you the old line about Linux being too hard to use. Point them to “Interview with a Grandmother” (www.linuxjournal.com/article/6562), in which Joe Klemmer talks to his mom about her experience with OEone's HomeBase Linux system. Not only is it so easy the proverbial grandmother can use it, this real grandmother uses her computer “ever so much more than before”.

Back in early January 2003, senior editor and business reporter Doc Searls prognosticated Linux and open-source events for 2003 in “Which Major PC Vendor Will Sell Desktop Linux First?” (www.linuxjournal.com/article/6548). As the title indicates, Doc feels this is the year Linux on the desktop will show up in a major way, thanks to the support of some major vendors. Is he right? Also, be sure to check out the predictions and comments offered by readers at the end of the article.

Finally, it wouldn't be *LJ* if we did not have at least one security article. In “Security with PHP Superglobals” (www.linuxjournal.com/article/6559), David

Lechnyr describes his desire to make after-the-deadline on-line reservations for ski equipment and how easy that turned out to be thanks to the site's use of GET statements. If the site had used PHP superglobals, which allow users "to specify which variables received by a specific method should be used", it would have been more secure, but he wouldn't have had skis waiting the next morning.

Just like having a music collection with a little Ella, a little Hank, some Buzzcocks and the essential Who is a good thing, so is having a web site with a little bit of everything—I mean, you'd rather be eclectic than boring, right?

Remember to check the *Linux Journal* web site often; new articles are posted daily. If you want to write an article for us, drop a line to info@linuxjournal.com.

Heather Mead is senior editor of *Linux Journal*.

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

[Advanced search](#)

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

Advanced search

Best of Technical Support

Various

Issue #108, April 2003

Our experts answer your technical questions.

Root Partition Mysteriously Read-Only

I have a Red Hat 7.2 system with two physical drives, each partitioned with three logical volumes. I have done something that caused my main root partition to be read-only. I've checked the directory permissions, and they are fine. But if I try to **vi** or touch a file in root or anywhere in a root subdir, I receive **touch: creating 'test': Read-only file system.**

—Jeff Manning, Jeff@VMWorks.com

Make sure you are truly on the root partition:

```
# cd /  
# df .
```

The output should include a Mounted on column with a value of /. If it truly is mounted read-only, it is most likely because you had some filesystem error. You should fix it before you mount the partition read-write:

```
# fsck /dev/hda2
```

Replace /dev/hda2 with the entry under the Filesystem column in the output of df. Finally, do this to mount the root partition as read-write:

```
# mount / -o remount,rw
```

—Christopher Wingert, cwingert@qualcomm.com

Switching to GDM from KDM

When my machine boots up in X, I get back what I think is the KDE login manager. I would prefer to use GNOME's GDM, as it has better features and is more customizable. How do I make GDM the default X login manager?

—Gordon Baldwin, gordonbaldwin@bigpond.com

Log in as root and edit the `/etc/sysconfig/desktop` file, changing:

```
DESKTOP="KDE"
```

to:

```
DESKTOP="GNOME"
```

—Keith Trollope, ktrollope@san.rr.com

Upgrade Makes Printers Disappear

I recently upgraded StarOffice from 5.2 to 6.0, and my system can no longer access the network printers. The printers are HP LaserJets with JetDirect cards, and they have their own IP addresses. Using spadmin outputs only to a default printer; there is no obvious way to specify an IP address. Looking at `psprint.config` shows a location parameter but offers no documentation saying how to set up an IP printer.

—Murray Zangen, murray@nj.com

First, add the printers to the system using `printtool` and specify JetDirect printer. Then, in `spadmin`, add the printer definition for StarOffice. When you get to the print command window, specify the printer in the command, for example:

```
lpr -P hp3
```

—Keith Trollope, ktrollope@san.rr.com

Emergency LILO Tip

This is a follow-up to the bulletproof backup/boot question that appeared in Best of Technical Support in the January 2003 issue. LILO knows about the `-R` option, which basically means “install this boot entry just once, then use default”. You may, at an early boot stage, install your emergency mode with **lilo -R**. At system shutdown (when all went well, if you are picky after remounting / as read-only) you may re-install your normal mode. So, if the box crashes

unexpectedly (or goes down quickly from, say, a forced reboot with Ctrl-Alt-Delete), you come up in emergency mode.

—Tomas, mlavergne@cfl.rr.com

Connecting to a Microsoft Network

How do I stick a Linux box into my Windows peer-to-peer network? I have an existing Windows peer-to-peer network that includes one Windows XP machine and three or four Windows 98 machines. They have fixed addresses in the 10.1.1.1-10.1.1.14 range. All the machines are completely shared, and there are no passwords; I'm the only actual user most of the time. How do I make the Windows machines see the drives on the Linux machine and vice versa?

—J. G. Owen, owen_labs@worldnet.att.net

Samba can help with this task. Setting up Samba this way can be confusing, however, because the documentation doesn't make it clear from the first step that there are actually two halves to this task. In Windows, client and server functionality is the same. Once the network is up, you can browse other systems, and they can browse yours. With Samba, each side has to access the other. Once Samba is installed, it takes only a little work to access other systems. You should be able to use the `smbmount` command to mount a share from another system. In fact, if you have other name resolution methods correctly installed, such as DNS, you generally don't need to run `nmbd` and `smbd`. For permanent shares, you can add a line to `/etc/fstab` that is similar to the following:

```
//machname/share /mountpt smbfs &username=xxx,password=yyy 1 1
```

You do need to set the Linux machine up as a server to enable other machines to access it. This is done by editing `smb.conf` to define the basic server properties as well as the shares to create. Then run `nmbd` and `smbd` to provide those services. The trick to avoiding sometimes-messy PDC-related work is to compile Samba such that it supports your normal password services. You can then add users to your system normally and use the `smbpasswd` utility to create the file Samba actually uses. It's a relatively manual process (described in the "Unofficial HOWTO" at www.samba.org), but it does do the trick. If you want to browse as well as share, make sure you add the `guest` or `no-name` account.

—Chad Robinson, crobinson@rfgonline.com

Various versions of Windows are a bit touchy with the Network Neighborhood. The best thing to do is edit `/etc/samba/smb.conf`. Make sure the "workgroup = line" matches your workgroup. Add a "netbios name =" for the name of your

machine. Depending on your installation, the default passwords will be picked up from /etc/password.

—Christopher Wingert, cwingert@qualcomm.com

I Have No Inetd and I Must...Telnet?

inetd on my system, Slackware 2.0.28, stops on occasion, barring access from telnet. Is there an easy way to automatically restart inetd if it dies?

—Mark Johnson, Mark.Johnson@InfoHarvest.ca

You're running an extremely old version of inetd, and some versions did have stability problems. Several solutions to your problem are available. If it is possible to upgrade your system, newer versions usually work fine, and you'll receive other benefits as well. If you'd like a replacement, try such alternatives as xinetd or daemontools, both of which are quite stable and add some features to the mix. However, if your cron service is trustworthy and you don't need to restart the service the second it dies, you might try running the following script every five minutes or so from cron:

```
#!/bin/sh
ISINETD=`ps ax | grep inetd |
grep -v grep | wc -l`
if [ $ISINETD != 1 ]; then
  /usr/sbin/inetd
fi
```

—Chad Robinson, crobinson@rfgonline.com

We've said this before, but please replace telnet with OpenSSH to avoid exposing your passwords and other sensitive data to the network. Packages for the OpenSSH client and server are available for all the Linux distributions, and compatible clients are available for every common platform. SSH is as easy to use as telnet, and it automatically encrypts your connection.

—Don Marti, info@linuxjournal.com

Setting the IP Address on SuSE

I'm operating SuSE 8.0 on my IBM ThinkPad 600E and am trying to connect to our LAN server at work. How do I configure TCP/IP so it automatically recognizes the addresses?

—Layla, satchumwatch@netscape.net

Run YaST2, the SuSE setup tool, and go to the Network address setup screen. Select Automatic address setup (via DHCP) to use a DHCP server if one is

available, or select Static address setup and fill in an IP address and subnet mask to set the address manually.

—Don Marti info@linuxjournal.com

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

[Advanced search](#)

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

[Advanced search](#)

New Products

Heather Mead

Issue #108, April 2003

Acceleron64 Opteron Servers, SCOoffice Server, iSystem Enterprise 3.0 and much more.

Acceleron64 Opteron Servers

Einix announced a new family of 64-bit dual processor AMD Opteron-based rackmount servers called the Acceleron64 series. The first offerings are the A1840, an enterprise-class server, and the A1820, a carrier-class server. The A1840 is a 1U server with support for four hot-swappable hard drives. RAID 5 are available, and Fibre Channel and iSCSI SANs connectivity to external storage arrays can be added. The A1820 is a 1U server with a hot-swappable 350W+350W redundant power supply. Both servers offer support for up to 12GB of DDR333 ECC memory, an Ultra 320 SCSI controller, dual GB Ethernet interfaces and hot-swap fans.

Contact Einix, 1313 North Milpitas Boulevard #110, Milpitas, California 95035, 866-883-4689, www.einux.com/about/acceleron64.php.

SCOoffice Server

SCOoffice Server, from The SCO Group, is a backoffice software suite for small- to medium-sized businesses. SCOoffice includes SCO Linux 4.0, SCOoffice Mail Server 2.0 and SCOoffice Base Server. SCO Linux 4.0 is based on the UnitedLinux 1.0 distribution. The Mail Server is preconfigured to support mail, address book and calendar features of various mail clients, including sendmail, Eudora and Outlook. The Base Server provides a graphical interface for configuring printers, file sharing, networks and other shared resources and utilities.

Contact The SCO Group, 355 South 520 West, Suite 100, Lindon, Utah 84042, 801-765-4999, www.sco.com.

iSystem Enterprise 3.0

MetiLinx, Inc. has released iSystem Enterprise 3.0, a software suite for adaptive infrastructure management. iSystem connects real-time analysis of system performance across every layer of the data center to tools that can redirect transactions, improving system availability and performance. Version 3.0 provides tools for server virtualization, IT cost allocation, database synchronization and replication and easy system integration. iSystem 3.0 automates many system management tasks, enabling multiplatform environments to self-manage and self-heal.

Contact MetiLinx, Inc., 999 Baker Way, Suite 410, San Mateo, California 94404, 888-399-5469, www.metilinx.com.

Appro 2U Server Series

Two new server series are available from Appro Computers, one using Xeon processors and one using Athlon processors, both using a 2U rackmount form factor. On the Xeon side, the 2224X is available with dual Xeon processors, four SCSI or IDE hard drives, up to 12GB of DDR, ECC memory (PC2100) and two 10/100 Ethernet ports. The 2228X system comes with eight SCSI drives. The Athlon machines offer dual Athlon MP processors, up to 4GB of DDR, ECC memory (PC2100) and two 10/100 Ethernet ports. The 2224 server comes with four SCSI or IDE hard drives, while the 2228 features eight SCSI drives.

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

Astaro Security Linux v4

At LinuxWorld NYC, Astaro Corporation announced version 4 of Astaro Security Linux, its all-in-one internet security software. Astaro Security combines a firewall/VPN gateway with software for spam filtering, content filtering, URL blocking and virus protection. New features for Astaro Security v4 include VLAN and WLAN support, virus protection for POP3 e-mail accounts, heuristic spam blocking and increased Radius and LDAP support for local and remote user authentication. The VPN authenticates via IPsec and preshared keys, and its PPTP offers 128-bit encryption. VPN throughput is up to 115Mb/second, and firewall throughput is up to 735Mb/second with a 1266MHz processor.

Contact Astaro Corporation, 67 South Bedford Street, Suite 400W, Burlington, Massachusetts 01803, 781-229-5880, info@astaro.com, www.astaro.com.

MontaVista CEE

MontaVista Linux Consumer Electronics Edition (CEE) is an embedded operating system and cross-development environment for consumer electronics devices. It features dynamic power management, enhanced filesystems, development tools for system performance tuning, processor and peripheral support, cross-development tools for system and application development, and hundreds of deployable utilities, libraries, drivers and other runtime components. Distributed in both binary and source formats, CEE provides real-time functionality and multiprocess and multithreaded support that can be used in devices such as mobile phones, set-top boxes, digital televisions and automotive telematics.

Contact MontaVista Software, 1237 East Arques Avenue, Sunnyvale, California 94085, 408-328-9200, www.mvista.com.

Umigumi

The OpenBrick Community introduced a new project, named Umigumi, that aims to simplify the installation and the distribution of OS installers by using Flash memory cards. Based on user input, Umigumi creates compact Flash cards with bootable code. The card can then be inserted into a different system to reconfigure that system as a router, firewall, VPN, OGG player, print server, thin client and so on. Originally designed to work with OpenBrick, Umigumi can support any embedded hardware platform, including PowerPC, StrongARM/XScale, Mips/Mipsel, SuperH, SPARC and 68K, running open-source operating systems.

Contact Umigumi, www.umigumi.org.

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

[Advanced search](#)

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