

Network or not work?

Networking with Linux is not only very popular, but also very important. Whether you use it from an IT services point of view, providing firewalls, print/file sharing, applications – or from a user point of view – setting up a home network, or just accessing data from the Internet. Here at Linux Format we operate on the principal that, whatever it is you're doing, you could probably do it better, so this month we're pleased to offer a gaggle (or whatever the correct collective noun is) of tutorials and features on the subject.

Finding new ways to run your network can be financially as well as educationally refreshing. For many people thin clients might be the alternative approach that makes sense, which is why we have a six page feature on the whys and wherefores of the technology, including a handy guide on setting up LTSP.

If the extent of your network is merely a desktop and a laptop, there are still some

networking tips to learn. Check out our special What on Earth on the new generation of networked filesystems, including one that will let you work on your shared files even when you're not connected.

Wireless networking is becoming more affordable, and achieving it under Linux isn't as hard as you might think, as you'll discover in the tutorial on page 68. If that gives you security nightmares, why not investigate VPN tunnelling courtesy of Dan DiNiccolo. And you certainly don't want to miss the start of a new series investigating Apache, kicking off with a very interesting install guide, that may teach you a few other tricks too.

And if you don't give a hoot about connectivity, there's plenty more this issue. Don't miss our DVD player roundup, our look at arcade game emulation and our host of programming tutorials. And the reviews, Help Wanted, our Answers section, Hot Picks...



Nick Veitch EDITOR

LINUX FORMAT

Aims of the magazine

Linux Format is a magazine dedicated to Linux and the Open Source community. We aim:

- » To provide the most accurate, unbiased and up to date information on all things Linux.
- » To promote the use of Linux in business and the home, for servers and on the desktop.
- » To support the Open Source community by providing a resource of information, and a forum for debate.
- » To help all readers get more from their Linux experience by providing insightful and useful tutorials.

LTSP – would thin clients suit your network needs? Read our overview and find out! p48

Which DVD player software is right for you – we've tested them all p38

Jeremy Allison talks exclusively to us about Samba and the future for Open Source p62



38



62



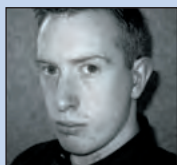
48

Meet Linux Format's team of writers...



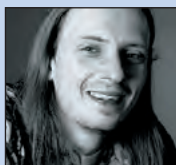
Richard Smedley

Taking a break from foraging for weeds to boil, our Prod Ed is allowed out to chat to Jeremy Allison



David Coulson

Our Answers guy is a networking and security guru with plenty of sysadmin experience.



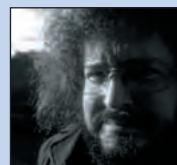
Richard Drummond

As well as writing our Java series, Rich co-ordinates most of the reviews in the mag.



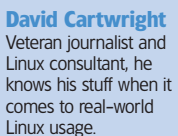
Jono Bacon

Founder of Linux UK and KDE developer, Jono is studying Multimedia at Wolverhampton Uni.



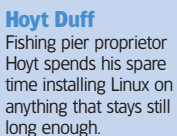
Charlie Stross

Master of Perl, Charlie has been writing about Linux for more years than anyone can remember.



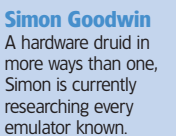
David Cartwright

Veteran journalist and Linux consultant, he knows his stuff when it comes to real-world Linux usage.



Hoyt Duff

Fishing pier proprietor Hoyt spends his spare time installing Linux on anything that stays still long enough.



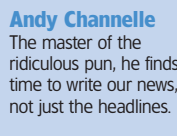
Simon Goodwin

A hardware druid in more ways than one, Simon is currently researching every emulator known.



Brian Long

Long time Delphi genius, Brian is also a dab hand with Borland's Kylix.



Andy Channelle

The master of the ridiculous pun, he finds time to write our news, not just the headlines.

Contact us

Letters for publication:
lxf.letters@futurenet.co.uk

Subscriptions/back issues:
linuxformat.subs@futurenet.co.uk

Technical help/Ask the Experts:
lxf.answers@futurenet.co.uk

Disc problems:
lxf.support@futurenet.co.uk

General enquiries:
linuxformat@futurenet.co.uk

Website: www.linuxformat.co.uk

Or send your letters to:
LINUX Format, Future Publishing, 30 Monmouth Street, Bath, BA1 2BW
Phone: 01225 442244
Fax: 01225 732295

More contact info on page 114

CONTENTS

LINUX

FORMAT

LXF26 April 2002

Welcome to another jam-packed issue of Linux Format, your guide to all things Linux!

DVD player Roundup

If you want to watch films while you code, find out which DVD player is the best for the job, and which can view the most file formats.



38

What on Earth...

... are the new network filesystems? Protect your shared data from network outage and synchronise your desktop files with your laptop.



54

Jeremy Allison

We interview the co-creator of Samba and find out his vision for the Windows-compatible file sharing system many find essential.



62

COVER FEATURE

SUPERIOR NETWORKING

Whatever your network set-up,
we can help you do more:

Linux Terminal Server Project **48**

Wireless networking **68**

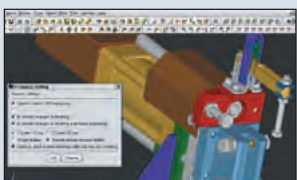
Apache tutorial **76**

VPN with FreeS/WAN **96**

Save money and subscribe to Linux Format. See page 100 or phone 0870 444 8645

»» REVIEWS

28 VARICAD
2D and 3D design made easy with this competent engineering CAD package. Is it the bargain that the price suggests?



31 SORCERER LINUX
Cast a spell over your PC as this distro provides complete control over your installation.

32 ACRONIS MIGRATE EASY
An easier way to upgrade your hard drive with this disk mirroring tool. Find out if it's worth the expense over standard Linux tools.



33 OMNIS
Database and web development made easy with the latest version of this visual RAD suite.

34 REVOLUTION
A new RAD platform based on a successful old visual programming language.

36 BOOKS
The book of Zope; The Linux Cookbook; Open Source Enterprise Systems & Java Tools for Extreme Programming.

»» TUTORIALS

68 WIRELESS NETWORKING
Untangle yourself with our guide to setting up 802.11b wireless networking hardware under Linux.

72 KPRESENTER
Show off your slides or stun the boardroom with KDE's powerful presentation software

76 APACHE
First part of our guide to setting up the most popular web server.

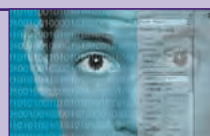


80 CVS
Concluding our look at CVS with a guide to setting up your own CVS server

84 PERL
Continuing the look at inter-process communications with a server for man pages.

88 JAVA
Object serialization – keep objects persistent between your program's executions.

90 KYLIX
Sort out errors with built-in exception handling.



96 VPN
Using FreeS/WAN to keep your company's data traffic secure between far-flung locations.

»» REGULARS

NEWSDESK	6
MAILSERVER	16
HELP WANTED	20
ANSWERS	22
HOT PICKS	43
USER GROUPS	110

»» FEATURES

NEW YORK EXPO	12
Linux in the big apple	
ROUNDUP – DVD PLAYERS	38
Movies on your Linux desktop	
EMULATORS	58
Put the blame on MAME	
JEREMY ALLISON	62
We interview the Samba king	

Coverdiscs

A DVD or 2 CDs packed full of the latest Linux goodies **102**



Security is the big theme this issue, reflected in the choice of Trustix Secure Linux as the main distro. *Kylux 2 Open Edition* and three video editing apps will keep you productive, and our music software will keep you entertained.

Please read the coverdisc instructions starting on page 102 before installing from the coverdiscs!



Newsdesk

Handhelds, workstations and server news; Free Software coders honoured; ridiculous patents proposed whilst Freedom Network releases source and Novell joins Liberty Alliance; and, gaming news, of course.

Sharp pushing Zaurus into the enterprise space

Sharp making alliances for Zaurus

Hot on the heels of its official release, the Sharp Zaurus SL-5000D gets its first update in the form of a new Flash ROM Image. The company says this upgrade is pretty much the equivalent of a beta ROM image for the SL5500. The upgrade finally adds IrDA communication to the device and also adds a voice recorder and audio-in driver, a backup/restore tool, an Internet wizard and IMAP4 support for the integrated email client. There have also been a number of tweaks and improvements in the core PIM applications.

SQL experts Sybase have teamed up with Sharp to "introduce initiatives to fuel development of Java-based mobile enterprise applications". Or, in non-marketing speech, both companies will be offering special packages to developers to encourage the development of 'm-commerce' applications for the Zaurus based on Sybase's *iAnywhere* platform. As well as offering a \$99 subscription to the SQL Anywhere Studio Program (which includes a bundle of software that would usually set you back \$2000), developers will qualify for a reduction on the development version of the handheld itself. Buying in to the programme will also give you access to development documentation, white papers and sample code.

Rob Veitch, director of business development at *iAnywhere* Solutions, said he thought *iAnywhere* developers would be anxious to port their applications to the Linux-based PDA.



You'll soon be able to go to Handango.com to buy (and sell) applications for the Sharp Zaurus SL-5000D.

"We feel that developers will be excited by the robust features of the Zaurus that are well-suited for the mobile environment," he said.

Isaac Ro, analyst with the Aberdeen Group, said that as the PDA market was growing so rapidly, it was important for Sharp to get its 'enterprise solutions' right from the very beginning. "By 2005, the market for PDAs will approach 40 million units, the majority of which will be sold into the enterprise. Against this backdrop, Sharp's partnership with *iAnywhere* Solutions demonstrates a clear understanding of the growing

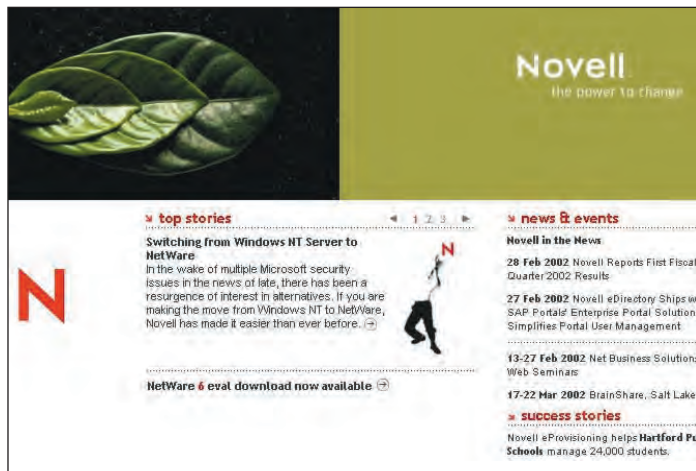
market demand for Java-enabled mobile enterprise solutions, and bodes well for both parties," he said.

Handango

Sharp have also announced a deal with handheld and wireless publisher Handango, to provide an online shop where users can buy new apps and utilities for their spiffy new hardware. Independent developers can sign up to have their apps hosted by the store which will initially be on the Sharp website, but may soon be accessible from Handango.com - giving you a potential audience of millions!

Handango's software partner program offers developers the opportunity to sell their wares on the international market, without having to worry about secure online sales, credit card validation or payment collection; the online store also provides detailed information on who is buying your software. However the service comes at a price - which is currently 30% of revenues - but, the company says, a developer has nothing to lose in offering their product through the online store: "if we sell nothing, we earn nothing."

<http://handango.com/Partner.jsp>



Novell joins the ranks of the Liberty Alliance.

Open authentication

Novell backing for Liberty Alliance

Network giants Novell have thrown their weight behind the Liberty Alliance, the group that hopes to challenge Microsoft's Passport in the online identity and authentication business. This new member gives the fledgling group much needed expertise in online identification and this, coupled with AOL-Time Warner's marketing abilities and Netscape's Directory Services, should make for a robust system. The Alliance has also gained the backing of infrastructure companies such as Hewlett-Packard and Cisco, as well as 'customer-facing' businesses like Intuit and the Bank of America.

As well as announcing the arrival of

Novell into the fold, the Liberty Alliance also welcomed another 11 members and said they intended to release "initial specifications for decentralized user authentication" by the middle of 2002.

"The Liberty Alliance is committed to creating an independent, open standard for user identity that puts the user first," said Eric Dean, chairman of the Alliance. "This can be accomplished by making it clear to users how their personal information is being used, giving them control over how it is handled, and providing the convenience that will result from companies working together."

www.projectliberty.org

HP's intent

HP launch Linux workstation

Hewlett-Packard is dipping

another toe into the Linux waters with the commercial launch of a pair of Linux-based workstations. The x1100 was initially release back in January as a Windows-only project, but the new edition will come with Red Hat preinstalled. HP says this is a continuation of their three-pronged strategy for operating systems (Windows, Linux and HP-UX). The basic x1100 model is built around an Intel processor and has a launch price of just over £1000, and a follow up (the x2000) should be unveiled in mid-March. HP's Linux workstations are currently making waves in



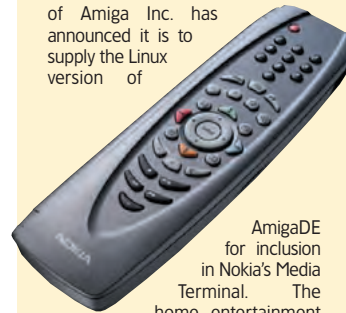
Hewlett Packard hope to take the animation world by storm with their Linux workstations.

Hollywood, with Dreamworks SKG 'buying hundreds' of the machines after successfully testing the system on the Mike Myers blockbuster, *Shrek*.

www.hp.com/workstations/

NEWSBYTES

■ The latest incarnation of Amiga Inc. has announced it is to supply the Linux version of



AmigaDE for inclusion in Nokia's Media Terminal. The home entertainment device, which has facilities for digital video broadcast (DVB) reception, streaming media and 'Net access, has had a pretty low profile in the UK and USA, but the device has actually been on the market in Sweden for a couple of months. It's expensive, but early adopters have reviewed the device positively.

■ Lindows.com have responded to the lawsuit slapped on them by MicroSoft, who claim the name is too close to Windows for comfort. CEO Michael Robertson told British news site The Register that the term windows had been used in connection with computer user interfaces before the launch of MS Windows in 1983. "We want to focus on building our product and not fighting in court, but we also think it's important to stand up to the bully on the first playground encounter otherwise he'll chase you home every day," he said.

■ Apple has launched a developers contest to find the best open source application port to OS-X.

■ Walmart (which owns Asda in the UK) may become the first big outlet to offer OS-free PCs for their more technically savvy customers. The world's biggest retail company says the products are intended for people with old Windows licenses or the desire to use an alternative OS. Cynics have suggested the range (which should start at about £399 for an entry level machine) is just a way to generate full price Windows XP sales...



■ It seems to have been imminent since God was a teenager, but Debian Planet claims a final version of Debian 3.0 is ready for release. As we go to press, they're just squashing the final bugs. "We've become accustomed to the Woody release process plodding along. But once base and standard are releasable, it has the potential to accelerate faster than you might expect"

Jono Bacon

The founder of UK Linux, KDE developer and all-round nice guy, Jono Bacon is studying at Wolverhampton University.



COMMENT

A charitable call to arms

“ This month I have been pondering about the people who are really affected by Linux. I am not talking about the tech-savvy developers, the cost reducing IT managers or the freedom fighters – I am talking about charities.

Charities are organisations where every penny matters, and reliability is essential; IT solutions are not just necessary costs, but can affect the lives of many people. A quick look at the Charities Commission website and registered charities' budgets gives a clear indication that many charities are under-funded and need all the help they can get.

Cue Linux, the free platform with little risk of the vendor lock-in that is associated with other solutions. It strikes me that a low cost Linux PC, complete with XFree86, desktop environment, office apps, web development tools, accounting software etc., would be a complete solution for a charity – add to this support by Linux User Groups and we have made a real difference.

The money saved can go into other more useful and effective areas; even if applications are purchased for Linux, the OS is still free and lock in is avoided. With the increasing ease of use of distros such as Mandrake, Lycoris and Xandros, Linux could be a serious contender to help charities.

Usually in my comment pieces I just express an opinion, but this month I am going to do something a little different – I would like to encourage each and every one of you to contact your local charities and explain how Linux can help them. I also encourage LUGs to organise demonstrations of Linux, and Installfests. I would like to hear about how you all get on, so email me with your experiences and use the LinuxFormat forums. Good luck!

Misunderstanding, not flamewar

GNOME and .NET – handbags at dawn

A bun-fight has broken out over the future of *GNOME* between two of the project's leading lights and a tech news service. However, it appears to have been 'all a misunderstanding' between Miguel de Icaza – who appeared to say .NET is *GNOME*'s future; Richard Stallman – who appeared to say 'over my dead body' and The Register – which 'took remarks out of context'.

De Icaza suggested that .NET was the 'natural' upgrade for *GNOME*, and he hoped that *GNOME* 4.0 would be based on .NET. "A lot of people just see .NET as a fantastic upgrade for the development platform from Microsoft,"

Stallman appeared to know nothing of this until asked about it at a

conference at Porto Alegre, Brazil. "I didn't know he was doing that," said Stallman, before suggesting the Ximian frontman should "explain himself" to the Free Software community.

De Icaza said there was no intention of basing future versions of *GNOME* on MS technology, and stressed the fact that no decision could be taken by him alone on the project's direction. "I am not the *GNOME* foundation or control *GNOME* like Linus controls his kernel, I am just its founder and a contributor," he said in an open letter to the community, adding that developers were hard at work completing *GNOME* 2.0 and that 4.0 "is not planned ... is a non-existent project."

A later re-rebuttal from Stallman



Context is everything when a remark can set brother against brother.

said that, though the quotes attributed to him were correct, they were taken out of context. "We will continue

wanting to support a wide variety of languages, and it is unlikely to make sense to rewrite *GNOME* into C#."

LinuxWebWatch



www.linuxnetworkx.com



www.beowulf-underground.org



www.linuxclustersinstitute.org



www.gridcomputingplanet.com

Cluster's last stand

Computers always get smaller, right? When it comes to clusters, big is most definitely beautiful.

In 1943, IBM's Chairman Thomas Watson said that he believed there was "a world market for maybe five computers." If Clusterphiles are right, Watson was out by approximately four.

Clustering is a way of linking computers in order to harness the power of many as if they were a single device. Of course the cost, performance and adaptability of Linux has made it a prime candidate for the *de facto* clustering OS, and as such there are many sites devoted to building and managing Linux clusters.

As well as flogging software

management solutions and some damn sexy hardware, Linux Network (www.linuxnetworkx.com) has a basic tutorial section which offers browsers a potted history of the technology and examples of its potential uses. It's got about as much depth as Kelly Brook, but is a good place to start if your cluster knowledge is minimal.

At the other end of the scale is The Beowulf Underground (www.beowulf-underground.org), which has "Unsponsored and unfettered information on building and using Beowulf systems." As you'd expect, this

gets pretty complicated very fast, but is an excellent source of news, links and info about software for creating, managing and measuring the performance of your cluster.

The Linux Cluster Institute (www.linuxclustersinstitute.org) has been set up "to provide education and technical training for the deployment and use of Linux-based computing clusters to the high performance computing community worldwide." The group runs a series of workshops on clustering that are aimed primarily at computer scientists and engineers.

Grid computing is being used to search for the 'magic bullet' that may cure cancer, in Oxford Uni's Centre for Computational Drug Discovery's project, that corrals together the redundant cycles in over one million PCs, over the Internet. Grid computing Planet (www.gridcomputingplanet.com) has a decent FAQ about the technology, and a rundown of some of its success stories.

Both IBM and Sun are making lots of noise about grid computing and you can find out more at www-1.ibm.com/servers/eserver/clusters/ and www.sun.com/software/gridware/

NEWSBYTES

■ Open Magazine reports that the US Navy Oceanographic Office will be undertaking a nine month study to evaluate the use of Free Software in its "specialized, mission-essential products and services for the fleet"

■ The organisers of the Linux Standard Base (LSB) are reinstating their pilot certification program. The LSB defines a set of standards that will increase application compatibility across Linux distributions, with the intention of increasing usability and stability. Sign up for the pilot scheme at www.linuxbase.org/test/pilot/



■ SuSE have secured a 4.4 million cash injection from Munich-based venture capitalists AdAstra Erste Beteiligungsgesellschaft. SuSE has substantially restructured its business over the past few months "to participate in the rapidly growing Linux market," commented AdAstra's Hans-Christian Perle.

■ There has been a quiet launch for the latest incarnation of Java from Sun Microsystems. The Java 2 Platform (J2SE) features a number of improvements and much requested features including new I/O facilities (including scalable operations for both sockets and files) and improvements to all areas of the Java Foundation Classes (JFC). Sun's Graham Hamilton said the company had worked with a number of partners to define and develop the J2SE 1.4 specifications. The result, he said, was a "high-quality specification that represents the diversity of the Java community"

■ From the some-people-have-too-much-time-on-their-hands dept: Richard J Kinch has put together a hardware/software combination that uses Linux to control his swimming pool! Kinch has documented the project exhaustively (really) at www.truetex.com/poolcontrol.htm.

■ Local government officials in Denmark are examining a number of Open Source options after they discovered plans for a 30% price increase in Windows and Office 2000 licenses. Moving over to something like StarOffice could save cash-strapped councils up to 100 per user, every year.

■ Meanwhile the UK finally has a body to promote the use of Free Software at a national level, following the founding of the Association For Free Software (AFFS), during the Sheffield Linux Seminar (see p112). As they gear up for their official launch, volunteer help is called for at www.affs.org.uk



Max Payne's developers said a Linux version was 'highly unlikely.'

Windows games on Linux

WineX brings Max Payne to Linux

TransGaming Technology have made another breakthrough in their endeavour to bring the latest games to Linux. For the first time the company have managed to get a DirectX 8.0 game – the very enjoyable *Max Payne* – working under WineX. What's more the developers achieved this feat without the support of Remedy Enterprises, the game's creators, or access to the *Max Payne* source code.

TransGaming's Gavriel State said the entire development team were very excited about the progress being made with WineX. "While the game isn't 100% yet, these developments demonstrate how rapidly our team is able to add support for new multimedia APIs. As we refine our technology, gamers can expect to be

able to play more and more top-rated games on Linux."

Max Payne was high on the WineX subscribers' wishlist. "It was TransGaming's subscribers who told us to make this happen," says State. "Our subscribers asked us for DirectX 8 compatibility, and recently released, triple 'A' game, *Max Payne* is the first game that works with this support."

In the game you play the eponymous Payne who is framed for a murder he didn't commit and sets out to bring the killers to justice, despatching literally thousands of ne'er-do-wells in the process.

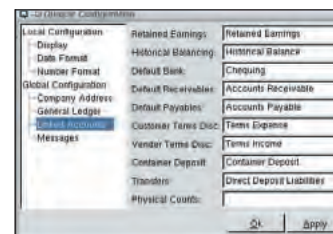
WineX is currently available for download by subscribers from TransGaming's web site, www.transgaming.com

Accounting software

Count on Linux

Canada Linux has just announced the availability of their latest accounts package, *Quasar1.1*. As well as being adaptable with the addition of modules to cover inventory control, remote access, shelf management and multi-store organisation, *Quasar* is also available in a 'Base Package' that is free for use on a single computer.

The Base Package appears to have all the tools a small business needs to manage the books, including financial statements, vendor billing, customer invoicing, receivable statements, cheque writing, bank reconciliation and reporting. At present *Quasar* links in



Quasar is a cost effective accounts package for small businesses.

with either the open source *Firebird* database, or with an embedded solution from Sybase – with further integration planned.

Download the free version, or an evaluation of the more sophisticated suite, from www.linuxcanada.com

Hoyt Duff

The author is one of 800 Hoyts living in the USA and runs a little fishing pier when he's not dabbling with his computers.



COMMENT

HTML email or not?

“Lately, a Unix-focused mail list I subscribe to has engaged in a heated debate about HTML-formatted email. The attitudes revealed seem to represent a microcosm of the current state of affairs in the Linux community; many discussions carry similar emotion-laden arguments.

Criticism of the use of HTML-formatted email focuses on three things: bandwidth and resources wasted; some Unix mail clients are unable to handle HTML; and HTML email is pro-MS – and therefore evil and unwanted. The other side argues that they use a particular mail client mostly for convenience in heterogeneous environments. Since my first use of Linux I had been with the anti-HTML camp until I examined the arguments.

Of course HTML uses more resources; so do *Enlightenment* and other apps, but no one relentlessly rails against them on that count. Besides, formatting text can convey additional info. I still do object to the animated icons I receive in a few emails.

Mail clients that can't handle HTML are simply broken: complain to the maintainer, write the code to fix it yourself or find a new app – but stop griping. Even the bewhiskered *pine* handles HTML.

Finally, my most important concern. Why does liking Linux (or any Unix) have to be about hating Microsoft? We live and work in a heterogeneous world, but some people just don't want to face that reality and get on with their business. One can be a strong 'nix advocate without getting so violently emotional, but perhaps the hate is a strong attractor to a sick and twisted few.

I have come to view non-productive arguments of this sort as juvenile. Linux has matured well enough to offer a commercially viable choice. Let's allow it to succeed on its merits.

"It's still cheaper than Office"

Sun to charge for StarOffice

Sun Microsystems have taken the unusual step of moving their previously free office suite, *StarOffice*, onto a more commercial footing. The company have announced they will make a royalty charge for the Windows and Linux versions of *StarOffice* 6, though native Solaris versions will remain free.

The company have stayed tight-lipped about the changes prior to the release, but the cost is expected to undercut the package's chief rival, MS *Office*, by a significant amount; to do otherwise would be both against the grain and commercially disastrous.

This will be the first time Sun have attempted to charge for the office suite since it was purchased from its original German developer.

The move has not been received with overwhelming scorn, as it may have a number of benefits. Firstly a

per-user licensing model would allow Sun to offer more comprehensive support for the application suite, and secondly it could stop the impression, which many corporate users have, that 'free' means 'worthless'. This latter reason has also been cited by Caldera as the basis for their change last year to per user licensing.

Of course, it's not all disastrous anyway as much of the *StarOffice* code is available for Windows and Linux through the *OpenOffice* project. The latest builds feature the same functionality, but is missing some proprietary technology such as the spellchecker.

Sun have made a number of recent commitments to Linux since the beginning of the year, including creating a custom Linux distro for Sun hardware; expanding support for the Sun.ONE architecture and improving



interoperability between Linux and Solaris. Sun are also extending their operations in the x86 market with a number of (multiprocessor) additions to the Cobalt range. "By mid-year, Sun will disclose details of its new family of general purpose, low-end Linux servers, including single and multiprocessing systems capable of running the thousands of native Linux and Java applications," the company said. <http://www.sun.com/products/staroffice/>

OpenOffice remains freely available, and has launched a marketing project: <http://www.openoffice.org/>

Virtual clusters

2000 Linux installs on one machine

If you think having Windows, BSD and three different flavours of Linux on your machine is impressive, you ought to see *Virtuozzo* 2.3 from SWSoft Inc.. The latest test result for the virtualisation software saw it handling up to 2000 instances of Linux on a reasonably specced Dell 8450 PowerEdge server. The tests were conducted using Red Hat 7.2.

SWSoft are aiming their product at web hosting and enterprise data centres.

"In a down economy where cost savings are paramount, issues such as server consolidation and increased performance of the data centre have never been so important," said Alexey Kuznetsov, who conducted the tests.

The package starts at \$250 and complete systems capable of scaling to 2,000 instances of Linux (based on Dell hardware) start at \$25,000.

www.sw-soft.com/en

Embedded Linux News



comes with the usual array of applications based on Trolltech's *Qt* PDA suite. <http://www.infomart.co.in/>

● The board of directors of the Embedded Linux Consortium (ELC) have 'laid the foundations' for building a unified embedded Linux platform specification. The decision has been a long time coming, but ELC Chairman and CEO of LynuxWorks, Dr. Inder Singh says the consortium "can now expand its activities into a crucial new area that the membership has been asking for: defining specifications for an open, multi-vendor embedded platform for the fragmented embedded market" <http://www.embedded-linux.org/>

● There are Linux-based PDAs coming out of the woodwork constantly now. This month's offering is called Kaii, from Infomart, and is aimed at the gap market between high-end Pocket PC machines and low-cost Palm devices. The device is available in mono or colour flavours and

● The New Internet Computer Company are targeting the grey surfer market with their latest offer. The Senior Explorer package will give America's senior citizen a NIC and unlimited Internet access for \$29.99 per month. The company intend to sell 25,000 NICs by the end of 2002 and a potential of 350,000 NICs by 2004. <http://www.thinknic.com/thinknic>



● Codeweavers have release version 1.1 of their *Crossover* Plugin which allows non-MS Operating Systems and browsers to access plugins designed for *Internet Explorer*. The latest version adds support for both *QuickTime* and Macromedia's *Shockwave Director* files, and will let users access all of the main IM services. <http://wine.codeweavers.com/>

Python creator recognised

Van Rossum gets FSF award

Python inventor Guido van Rossum has been given the fourth annual Free Software Foundation (FSF) Award for the Advancement of Free Software. Van Rossum received his accolade from FSF President and founder Richard Stallman at the Free and Open Source Software Developers' Meeting (FOSDEM) in Brussels.

In an interview with GNU-Friends.org, van Rossum said the award perplexed him because "I've always found it a natural thing to release my software as open source; in fact I've done this since I first started programming in the mid-seventies. So I feel I've just done what comes natural to me - not something particularly worthy of an award." He also heaped praise on the Python community for the "contributions I've received over the years."



Guido van Rossum is the fourth recipient of the FSF Award.

The other two finalists this year were L. Peter Deutsch who has brought robust postscript functionality to Linux with GNU *Ghostscript*, and Andrew Tridgell (*Samba* - see p62).

Pensacola beta: available for your computer-crashing pleasure.

Red Hat server specials

Red Hat's Pensacola is coming soon

The corporate face of Linux, Red Hat, has released a beta version of their server-orientated Linux product, Pensacola. This release is definitely not for mission-critical applications.

Over and above the usual goodies we've come to expect from a Red Hat distro, Pensacola introduces two new areas of functionality – clustering and a specially tuned kernel.

The clustering technology is based around Red Hat's respected *Piranha* core, but also includes 'new shared storage failover clustering' in which one node can physically block the power supply to another node for 'data fencing'. Improvements to the kernel include I/O scalability improvements, large memory tuning, multi CPU improvements and hyperthreading support.

Red Hat has been toasting the inclusion of Red Hat 7.2 in the Datamation Product of the Year Awards for 2001. The company's server product took the crown from Microsoft Windows 2000 Advance Server in the Network and Systems Software category. The server OS polled twice as many votes as its nearest rival, Macromedia's *Jrun Server 3.1*. Datamation said the accolade 'underscores the major shift under way in the enterprise. If the votes of Datamation readers – primarily IT executives with experience and buying power – is any indication, Linux software appears to have gained a permanent and growing role in the enterprise.'

<ftp://ftp.redhat.com/pub/redhat/linux/beta/pensacola/>

Anonymity for research purposes

Freedom Network source code goes wild and free

In the fall out of September 11, a number of encryption and anonymity services have faced being tarred with the 'terrorist' brush, and for many sites political harassment and bad press have led inevitably to closure. The Freedom Network was one such operation that closed its virtual doors last October, though Chief Scientist Ian Goldberg said the decision to fold had been taken before September.

Now the source code has been released into the wild under "an RSAREF-style license" for the benefit of academic research and other non-commercial uses. Freedom Network used a number of approaches – including encryption, complex routing and noise generation – to ensure the anonymity of its users. The main tarball can be downloaded from

www.codecon.org/goodies/freedom

Patent roundup

» **EUROLINUX** There's something fishy going on at the European Commission if EuroLinux, a group that campaigns to stop software patents, is correct. The group have seen a draft copy of the proposed directive on software patents. 'This same document', EuroLinux says, 'has been sent to a number of official EU representatives.'

The proposal seems to have been written or revised by patent expert Francisco Mingorance, who is also the Director of Public Policy for the Business Software Alliance (BSA), which 'represents the interests of large US publishers in Europe'.

The 27 page report says that patents 'act as an incentive to invest the necessary time and capital and it stimulates employment', and suggests that we all reap the 'benefits from the disclosure of an invention, which brings about technological progress upon which other inventors can build'. EuroLinux claim there is no need for extending patent law as software already receives adequate protection from copyright legislation.



Eurolinux questions the wisdom of letting an organisation so closely allied with the industry influence patent policy.

Matthias Schlegel, CEO of Phaidros – one of the companies supporting EuroLinux – said copyright "provides a simple and very efficient protection to the software economy", whereas patents tend to allow one company to "monopolize an idea of software or idea of business on the Internet, prohibiting other companies to use the same idea, even when it is implemented differently".

Stéphane Fermigier, CEO of Nuxeo, said that European publishers are probably unwittingly infringing on many of the 50,000 patents owned by big US developers such as Sun, IBM and Microsoft. "Instead of protecting software publishers, software patents create a tremendous juridical uncertainty and allow large IT companies to completely control the software economy, block innovation and block competition by prohibiting one software from being compatible with another."

» **BRITISH TELECOM** BT's claims to have invented the hyperlink took another blow as, on the eve of their first real hearing in a US court, Bob Berner – the self-styled 'world's oldest programmer' – claims to have come up with the concept of hyperlinks way back in 1960. BT's claim dates from 1976. Patent experts don't believe Berner's interjection will have much bearing on the case because BT is unlikely to win. Presiding U.S. District Judge Colleen McMahon said the wording of the patent application was positively archaic. "It appears that this technology was already outmoded by the time it was patented," she said.

David Cartwright

David Cartwright is an IT consultant who specialises in providing Linux systems and solutions.



COMMENT Embedded Linux OS

“ With Linux becoming more and more popular as a server operating system (let's not get into the discussion of whether it's ready for the desktop yet), many people's attention appears to be turning to its usefulness as an "embedded" operating system – the base-level code that you use to run applications on PDAs, mobile phones, digital watches etc. But is Linux really what you need?

The trick with embedded OSs is that one day your cooker will talk to your video, your TV will talk to your digital watch, and your mobile will integrate seamlessly with the answering machine and, while we're at it, the lighting circuits in your house. Although communications technologies such as Bluetooth have been slow to get off the ground, it's believable that intercommunication between dissimilar devices will become more and more desirable to the consumer – and so the manufacturers will want to make money by selling the gadgets that do what the consumer wants.

There are two ways this might be achieved. One is the MS way – be huge and do it your own way, thus providing *de facto* standards that the world adopts because it's too much hassle to do it any other way. The other is a more collaborative way, where design and development are perhaps less controlled but the ways of doing things are much more open and people all over the world can contribute their own bit within a commonly-agreed framework.

So what do we need for the ideal embedded OS platform? Well, it needs to be open, small, fast, inexpensive, versatile, standards-based and able to support as many of the standard protocols (for intercommunication, display technology, and so on) as possible.

That'll be Linux, then. ”



Mailserver

Share your opinions, right wrongs and demand justice by writing in to Linux Format. Drop us a line at: **Linux Format, Future Publishing, 30 Monmouth Street, Bath BA1 2BW** or email: lxf.letters@futurenet.co.uk

★ Letter of the month

This month's winner receives a copy of *Python & XML*, by Jones & Drake

An overworked theme...

What do pp 19, 59, 72 & 73 and pp 18, 43, 44, 73, 76 & 78 have in common? Some secret code? Old-style line noise? No! The first list are pages in the February 2002 LXF24, where someone had a theme set in the X session that makes text in *xterms* unreadable. The second are pages in the same issue that are just about readable, but still exhibit over-enthusiastic use of themes.

Please can you be more

restrained in future and remember that some readers have less than perfect eyesight. Even with glasses or contact lenses I, for one, can't read these screen captures.

Trevor Jenkins, *via email*

Yes. We are aware of the problem. It is a little harder to deal with than you think, because the printing itself is affected by the paper we use, as well as the quality, size and colour balance of the images, so it can be impossible to predict exactly how

such screenshots will appear before we print them. The backdrops were to make the images seem a little more interesting, but obviously this hasn't worked! Don't worry, we are working on this. However, in the meantime you can enjoy the star prize this month, *Python & XML* from Oreilly (Jones&Drake, ISBN 0-596-00128-2), which doesn't have many pictures in it at all!



available in Perl. There is just sooooo much free XML Perl stuff available that his article is nonsense. XML has been supported with free tools under Linux for ages. ASP is pretty much a MS invention (though it does run under *Apache* now), so there is not so much support for ASP in Linux. But what the hell, who needs it when you can do so much with Perl and CGI?

Ron McKeating, *Computing Officer Loughborough University*

Debian again

I was delighted to see that you included Debian 2.2r4 on the cover DVD of your magazine. Since Debian 2.2r5 has been recently released I hope you will consider putting that on the cover disk soon.

David Craig, *via email*

We don't have anything against Debian, and we'll be featuring new versions as and when it seems appropriate.

More distros

I don't want to step on the toes of those who like non-distro software (I like it too) but I'd like to make a plea for general distros on the CD. A lot of people must be in my situation who don't have either a DVD reader or a CD writer and not able to afford them yet (some of us are still making the best of some old hardware). It's nice to get the next version of a distribution when you start reading of problems with the current one.

What about saying that:

a) 3 or 4 CDs a year are reserved for distros and the rest for other software or

b) 3 or 4 issues a year have a second CD (this has happened once already).

Thanks for your recent articles

Mailservers

I think I may have the answer to Leon Stedman's problem with his mail server. It took me a year or so to discover why I could send mail to root but not send it back to my real users. Anyway the answer probably lies in his user account names, Leon & Pat.

Sendmail does not like capital letters (or at least this was the case a few versions or so back) so if you send mail to "Leon" it will not work. If Leon changes his accounts to "leon" and "pat" everything should be OK. And just to make sure that incoming mail for Leon or Pat gets to the right account, you should (assuming Red Hat/Mandrake distro) put

Leon: leon

Pat: pat

in */etc/aliases* and run *newaliases*.

Hope this helps him.

Rob Clayton, *Carlisle*

Our readers know everything. And

thanks to David Fussner who also wrote in with similar advice.

CLIphobia

I am currently using Windows 98 because it means that I manage to avoid having anything to do with MS DOS (where I do a great deal of



Redmond Linux – happy to share your disk with another OS.

damage), but would like to know more about Linux. I have read a "Dummies" title which seemed to suggest an awful lot of command line usage and I do not feel ready for this. I am beginning to wonder now if the idea is not too ambitious.

I read your review of the Mandrake 8.1 distribution but wonder if you have any thoughts on the Redmond Linux product which was recently released? Can I assume that either of these would share my hard disk with the existing Windows system?

Malcolm Stringer, *via email*

It depends what you want to do with your system. If you just want to use the provided applications, you may not ever need to use a command line. Pretty much every distribution you will have heard of will co-exist happily with Windows.

XML and Perl

Great Mag.

Message for Dave Cartwright: look at all the XML modules



Linux in your pocket – everyone's doing it.

on using older hardware, I'm using older Dell kit myself. Great mag otherwise, etc.

John Stevenson, via email

Now that we have 2CDs on the CD issue, we'll be able to provide distros more often, as well as other larger stuff that would take up too much of the CD to have considered previously.

Brief iPaQ

Your review of the iPaQ was somewhat brief.

I've recently installed Linux on a brand new 3850, so it is possible for a Linux newbie. Perhaps you could run an article on familiar and intimate on an iPaQ?

Jonathan Chetwynd, via email

It wasn't a review of the iPaQ, we were reviewing the Zaurus and mentioned other Linux PDA experiments in the process. There is much information available on installing Linux on iPaQs, but we may consider a brief guide. If I knew what you meant by 'familiar and intimate' there might be more chance of us covering it.

Wordpros for Linux

Thanks for the responses. I would like to respond to your comments on Linux word processors. I would love to use *AbiWord*, or *KWord* for that matter.

I looked at *AbiWord*, *KWord* and *OpenOffice.org* before deciding on using *StarOffice*. *AbiWord* is easier to use than *StarOffice* and seems to be nice and stable. I prefer the *AbiWord* interface to *KWord* (which is also coming along nicely), *StarOffice* and *OpenOffice*.

Unfortunately, both *AbiWord* and *KWord* lack footnoting. This is an essential feature for writing academic essays at my college and effectively renders both *AbiWord* and *KWord* useless. Footnoting is planned for *AbiWord* but not in the

immediate future, or the 1.0 release. Yes, there are work arounds but these are for technical people who know what they are doing, not for your average or computer-phobic user. It's also not right to give even experienced users a word processor to write essays for college work, and then tell them they will have to use a complex work around for a required function.

Thus, we are left with *StarOffice* or *OpenOffice*. I have recently upgraded the system to use *OpenOffice 641 C*. This solves the problem of saving to floppies in *StarOffice* but is not without its own problems. Primarily, the spellchecker is limited to American English, the styles dialogue box can't be turned off when it loads, and remembering and setting the zoom level has some glitches. Admittedly these last two items are minor. (I also prefer *OpenOffice* since it is Open Source software – unlike *StarOffice* – but when I have committees and users who care only about functionality and ease of use, the Open Source issue is often neglected.)

Hopefully these issues will resolve themselves in the near future. With any luck, next year's students will have a choice of word processors. The future of word processing on Linux looks bright even if I can't get exactly what I want immediately.

Perhaps this is an issue you could consider featuring in the magazine. Looking at the different word processors and what sort of application (academic, scientific, internal reports, letters, published

documents, etc ...) they are suitable for.

David Petticrew, Manchester

Thanks for writing back – I think we understand your needs a little better now. I hope the developers are taking note!

Lifeline

All I can say is I will never let my subscription lapse.

The reason: on my work laptop, that runs Windows 2000, I installed Red Hat 7.2 to co-exist with Windows. This to help with work related issues. I have installed a few versions of Linux in the past, with no problems. But never with *GRUB*.

The install went like a dream, fast and smooth. So far so good. I selected to overwrite the boot

Next year's students will have a choice of word processors. The future of word processing on Linux looks bright even if I can't get what I want immediately.

loader of Windows thinking it would place an entry into the *GRUB* loader.

Bad mistake, never assume: I should have known this but even the experienced forget things. So on boot up I was missing the option of Windows. Then I remembered the August *LXF* had a *GRUB* section in it. After digging around in the cupboard, a few minutes and swear words later, I emerge triumphantly holding my reward.

Within 5 minutes my laptop was up and running with Windows.

Thanks for a great mag, keep up the good and helpful articles.

Kevin, London

So let that be a lesson to you all – don't use your back issues to line the budgie's cage, you might need them again some day.

More upgrades?

I think your reply to Barry Snelson ("Stop the upgrades!", *LXF 21*) misses the point Barry was making.

You simply cannot dismiss the problems ordinary users have with installing new software by basically saying "oh well, you should realise when you are trying to install beta or development software that uses new libraries you don't have, so in future stick to a stable distribution

and stable packages" – and still expect Linux to grow in popularity among the general community.

My attempts at, for example, upgrading from *KDE 2.1.1* that came with my *MandrakeLinux 8.0* *PowerPack* distro, to what I assumed to be stable versions of *KDE 2.2* (*LXF 20*) or *KDE 2.2.1* (*LXF 21*), have failed because of missing libraries, etc. It was the same experience with *Video4Linux* (*LXF 20*). I admire the skills of the people who develop software but, like most ordinary users, I simply don't have the skills to confidently edit config files and make files; or easily figure out paths to what seem to me (as an ordinary user) to be obscure system files. I can't make sense of a "couldn't locate a 'GLUT' source" message and I don't have hours of free time to hunt down missing libraries.

The frustration I feel is not at my lack of skills – it's because I actually like Linux and *KDE* as my window to using and enjoying some great applications – but as a general community user of Linux my experience is not a happy one, >>



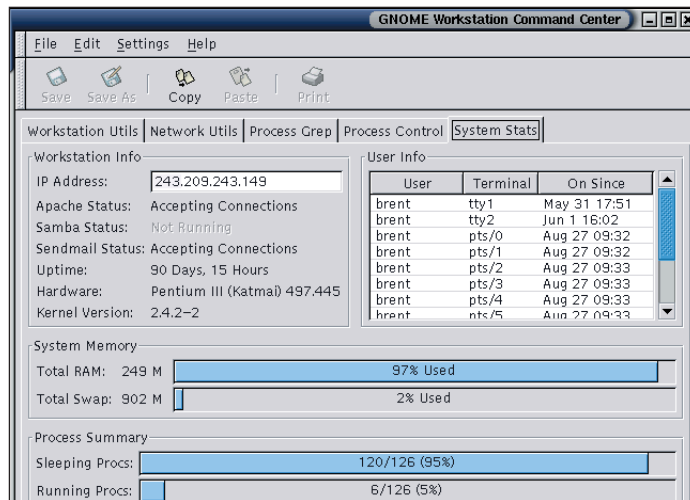
In case of emergency, open back issue (if you still have it...).



« because I usually can't get new or improved versions of applications to install successfully.

It seems to me that this is where MicroSoft has the edge. Much as I dislike Windows, I know that if I install a newly released or upgraded application for Windows today, I can be confident that it will easily install and run under my four year old version of Windows98SE. How and why it works is not something I need to understand because I'm a product consumer, not a product developer. But if I install a newly released or upgraded Linux application today I don't have confidence that it will easily install and run under my 12 month old version of Linux.

This takes me back to Barry's question – does the Linux community actually want Linux to be popular among the general community (and if so is it prepared to work together to ensure that we ordinary consumers don't have to become software developers to get



GNOME – needs a lot of libraries, so let your distro take the strain.

Minor patches may be released, but new features will be rolled into a later version a year or so down the line, which will cost you more to upgrade.

Now consider the equally fictional Open Source software *FreeOffice*. It is freely available. There are no manuals or support because there is no money

which is released on Linux, for example *Kylix*, or *Loki* games, you'll find that they pretty much install in the same way as would any Windows app – they have an installer and include any necessary system updates. This is fine if you reach a stage where your product is 'finished', but as we have said, most Linux software is in constant development.

That said, many Open Source projects do take the time to make comprehensive installation packages, including *KDE*, *GNOME*, *OpenOffice* and other projects. If you would rather wait for versions that will work with your system without you having to do anything, you'd be better off getting the update to your distro every three or four months.

I do understand what you are saying. Yes, it is inconvenient and troublesome to update half a dozen libraries just to install a new app. But that is really the nature of running such software. It isn't anything the developers can do much about, unless they didn't take advantage of new technology at all.

That said, if you run the "testing"

branch of Debian you can install all the latest Free Software *and* have all dependencies taken care of automatically. Many other distributions now also feature automatic updating packages to take care of this.

DMCA

As with any job well done, at the very least an "atta boy" is in line. So I felt I would drop one on ya'll. Its good to see that *Linux Format* has found its way across the ocean and into my local "Borders" book store. Yeah, the folks at Borders can't seem to get the GBP to US\$ conversion down correctly, but hey, its worth the few extra \$. It looks like your designers put a lot of time into the format, and your editors make sure the content is clear and to the point. Keeps my interest a lot more than your US competing magazines.

Anyhow, about your web page... it's beyond resurrection. I think the only other website that compares in its awfullness is the home page for SNORT. Please do something!!!

There is one thing that I would like to thank you all for is your articles on the DMCA laws. You would probably be terrified at how many Americans in the IT world have no idea that this law exists. To be honest, what makes it worse is that I found out about DMCA in a Linux mag from the UK. It's nice to hear at least some folk in the world are concerned about what goes on in the US, and are aware of the repercussion therein. Sadly, US media is not quite as... informed/brutal/on top of/concerned with the laws which go into effect as are European news agencies (yeah, I watch BBC). Anywho, it seems lobbyists/corporations determine what is newsworthy in the US, and therefore the lay person is

Entirely different models of software development. It isn't a question of making things harder for new users. It's just the way the software is developed.

new applications to run), or does it secretly like its exclusivity?

Murray Hawkins, via email

Thanks for your comments on this topic. I think you may have missed the point of my answer. Consider the fictional software *ProprietaryOffice*. It is a Windows application. It comes with manuals, support and installation help. It costs an amount of money.

to pay for them. It is released almost constantly – there are new versions every month or so, including bug fixes and new features.

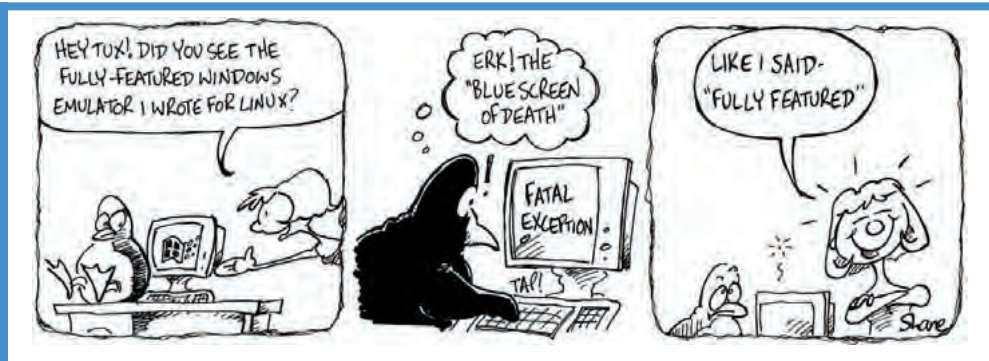
These are two entirely different models of software development. It isn't a question of making things harder for new users, it's just the way the software is developed.

If you look at proprietary software

Helpdex

By Shane Collinge

shane_collinge@yahoo.com





Shurely not...

I wonder how *KDE* exponent and *Linux Format* contributor Jono Bacon manages to do all his computer work and also be Adrian Smith, one of the guitarists in top UK rock band



Bacon

Iron Maiden.

Just compare the two pictures and you will see...

Chris Arkle, *Penicuik, Midlothian*

One of them is certainly an expert at string handling...



Maiden

completely unaware of how our rights are slowly but surely stripped away. Sad huh?

Well, I will continue to purchase this awesome magazine, and spread its word to my fellow co-workers. Keep up the good work!

Chris D, *Virginia USA*

We appreciate the kind words. Do you have any suggestions for the website? Most people seem to like it OK!

The whole issue of software patents and copyright legislation is something that effects everyone, but not many seem even vaguely aware of what's going on, even in Europe. I don't think it gets enough coverage even in the UK media.

Cracker snacks

First of all, I want to say that it is a good magazine. But there's one thing that is remarkable. On the CD of the January 2002 magazine, there's a tool called *Ettercap*. This tool can be used for different purposes, such as sniffing and

logging. These functionalities are handy for network administrators, but they are also easy to use for crackers. Therefore, I found it remarkable that a magazine is distributing these kind of tools.

As a member of a CERT (Computer Emergency Response Team) team, I have to say that everybody has a responsibility to keep the Internet as safe as we can. This counts for end users and ISP's but also publishers. Of course crackers or script-kiddies, but also network administrators, will find these kinds of tools on the Internet. But a magazine shouldn't stimulate these tools, by distributing via a CD.

Hope you will take this opinion in concern. Except for this mistake: Keep up the good work.

Carol Overes De Kwakel,

The Netherlands

I appreciate the point. If we ran a tutorial on cracking that would be irresponsible. But the main use of tools such as the one you mentioned

is to secure systems. Crackers will get these tools, or more likely, more specific ones designed for cracking, from secret FTP sites or through their chums on IRC channels.

Of course, we do not want to in any way encourage cracking, but not including useful network tools on the CD won't help any. Or are we wrong?

Well prepared

First of all, I'd like to say how nice it is to see some familiar faces. I used to read *Amiga Format* all those years ago when I still had my little A500 (sob). Imagine my surprise when I picked up *Linux Format* and saw the same cheeky, grinning people! Now, to business...

Viruses – afraid or prepared? Viruses should be treated like any software problem (be it a bug or whatever). The Linux community has a distinct advantage over the virus writers. Unlike with other (non-free) OSs, Linux users can become very involved with the community. (I imagine everyone who's been using Linux for more than a few months has posted to some Linux-related forum or other.) Consider a familiar example: whenever a new kernel release is issued, that is in some way buggy, floods of people the world over post the fact and, quicker than you can say “antidissestablishmentarianism”, a patch is coded and released. Now I would imagine the Linux community would react the same way to viruses. Some new killer virus is found, lots of people post to their favourite Linux newsgroup about it, and pretty soon someone has written the antidote. Hurrah for our brave GNU world.

I would also like to dispel a common myth, namely that Linux is hard to use. I have been using

Debian (apparently one of the most “hardcore” distros) for about a year now. Before that, I used Mandrake for a few months. I have to say that learning how to use Debian has been an absolute joy. Of course, there were a few hiccups during the initial months: getting used to *bash*, Linux devices, etc. All in all, Linux is great fun to learn and to use. Once you pick up a few important concepts, everything becomes much easier and you can start getting things the way you want them. That's the best part about Linux – it works exactly how you want it to.

That's my lot. Thanks for the magazine, I'll be taking out a subscription as soon as I know whether or not I can afford a DVD drive.

Steve Engledow, *Norwich*

P.S. Isn't it about time someone ported *AMOS* to Linux?

Cheeky faces! Blimey. How I will remember all those cheeky readers. Anyway, I think you have a point that the Linux community is better prepared to react to viruses – but I guess the problem is that reaction, for some, will be too late. Are our systems secure enough to minimise virus damage? [LXF](#)

Mailserver Hot Topics

We have introduced Mailserver Hot Topics to help gauge your opinions on the things that matter most. Please feel free to continue writing in on any subject you like (except the glues we use on the CDs, everyone's heard enough about that), but we would be extra keen to hear your views on the hot topic of the moment. Without further ado, next month's topic is: **StarOffice - are Sun right to charge for Linux versions?**

Submission advice

WHAT WE WANT:

- Letters about the magazine or Linux in general
- Constructive criticism
- Your opinions
- Concise points about relevant subjects

WHAT WE DON'T WANT:

- Technical question – direct those to our Q&A pages!
- Random abuse
- Nonsense rants
- 200 pages of meandering diatribe

WRITE TO US AT:

Linux Format, Future Publishing, 30 Monmouth Street, Bath BA1 2BW or email: lxf.letters@futurenet.co.uk



Answers

If you are really stuck and the HOWTOs yield no good result, why not write in? Our resident experts will answer even your most complicated problems!

Experts this month

Whatever your question is, we can find an expert to answer it – from installation and modern woes to network administrations, we can find the answer for you – just fire off a letter or email and it'll all be taken care of.

LXF answers guy
David Coulson
is a networking and security guru with plenty of sysadmin experience to boot.



Richard Drummond is an experienced programmer who can answer queries on a variety of subjects. A keen Debian user, he's also our resident Java guru.



Nick Veitch is the editor of the magazine, and answers your easy questions! Or indeed anything to do with *Grub*, *LILO*, *netatalk*, vi...



Trident graphics

Q I'm trying to install Red Hat 7.2 (from last month's magazine) on my new Toshiba 1800-814 laptop and the XFree86 GUI is not working. I have discovered it's because the graphic card (Trident CyberBlade) is unsupported. How can I work around this? I am quite new to Linux and would be grateful for any answer provided it's understandable to someone lacking in expertise. Thanks, *Robbie Thompson*

A This laptop is indeed very new, because no one has included any information about it at <http://www.linux-on-laptops.com/toshiba.html>. However, all is not lost, as XFree86 4.x supports the Trident CyberBlade video card, according to www.xfree86.org. Why Red Hat suggests otherwise is not clear. You may wish to manually select a video card, rather than have it probe for it.

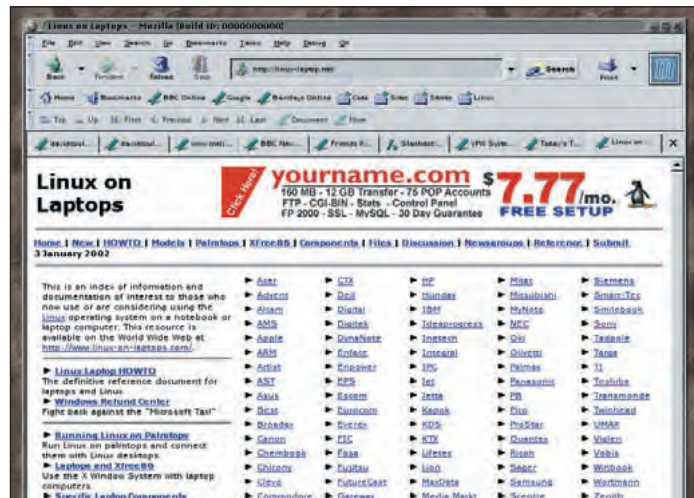
As the same driver handles a number of different Trident cards, you can, in theory, pick any one of them and it should function properly. If it still does not work, your best option may be to install Red Hat without X, then download packages for XFree86 4.2.0 from ftp.redhat.com then do **XFree86 -configure**, which should be a little more productive than Red Hat's effort.

USB problems

Q My wife has just told me off for spending far too much time on my computer attempting to get my USB scanner to work. She has a point and I hope you can help.

I am running the SuSE 7.2, 2.4.4-4GB-SMP kernel, on an ABIT VP6 motherboard (VIA chip set) sporting a pair of 1000 MHz PIIIs.

Having freshly booted into SuSE 7.2, **lsmod** shows that *usbcore* is



If you're having problems with a laptop, then the first place to look is the 'Linux On Laptops' site at www.linux-laptops.net

the only USB module that is loaded. As a result, *usbview* doesn't show any USB devices. **modprobe usb-uhci** causes *usbview* to show two USB hubs as well as an unknown device (I presume that this is my scanner).

All is not well however since the USB version, device class, device subclass and device protocol fields all show 00. After waiting 5 seconds or so, the unknown device will disappear if the *usbview* screen is refreshed.

After running out of other things to try, I compiled and installed the 2.4.10 kernel (with support for SMP), which you kindly supplied on December's LXF coverdisc. I naturally compiled USB support into the kernel and hey presto! Not only does my scanner appear in *usbview*, but I can also scan to my heart's content using *xsane*. Unfortunately, as a newbie, I'm not smart enough to make my printer or modem work under 2.4.10, but at least this proves that my scanner (which is an Epson 1640SU) and USB port function correctly.

This all leads me to believe that

the SuSE 7.2 2.4.4-4GB-SMP kernel has not been configured for USB support, but unfortunately SuSE were not able to offer any help. Come to think of it SuSE technical support hasn't managed to solve any of the problems I have experienced with SuSE 7.1 and 7.2, but that's another story.

A quick look around on the Internet leads me to suspect that there may be an issue with SMP, USB and APIC IRQ routing although, like I said, my scanner works fine with the 2.4.10 kernel.

I went for the 1640SU because I thought that I could always use the SCSI interface to get me out of trouble if the USB interface beat me – this could turn out to be a smart move, although I'm confident that the egg heads at LXF will be able to show me the error of my ways.

Anyway, time to bond with the wife and kids...

Hope you can help ;-)

Steve Roper

A SuSE, as with every other distribution under the sun, patch their kernels to the

moon and back, which is great as long as everything works, but not quite so useful if you are having problems. As USB works happily under 2.4.10, you may wish to work on getting your printer (which is presumably parallel) and your modem working correctly, rather than messing around with SuSE's kernel. If your printer is parallel, you're probably not compiling the 'Parallel Port' support within 'Character Devices' in the kernel, or you need to compile in the modules for parallel printers. As we don't have any log information, such as when *lpd* starts up, it's difficult to figure out why it doesn't work with 2.4.10.

The modem could be a little more interesting. If it's a regular serial modem, then you just need support for regular serial interfaces, but if it's one of those lovely Winmodems, then you'll need to head over to www.linmodems.org and download the appropriate driver, or copy across the module from the SuSE */lib/modules/2.4.4-4Gb-SMP/* directory and forcibly load it with **insmod -f**. 2.4.4 is a very old kernel, and 2.4.10 is quite out of date too, so you may wish to start afresh with 2.4.17, as USB is frequently upgraded in the kernel to support extra devices. You would think that SuSE would distribute functional USB modules with 7.2, but it could be down to a bug in the kernel with your particular motherboard, or a fault from their compilation procedures. Certain Apollo Pro motherboards, including the VP6, have APIC issues under Linux with SMP, which you can solve by including the line; **append="noapic"** in */etc/lilo.conf*, then running */sbin/lilo*, although with APIC it generally either works, or it doesn't, and many of the problems encountered with the particular motherboard chip set involved hard lock ups, rather than specific failures.

More USB

Many thanks for the interesting article on USB devices in LXF22. Armed with the knowledge gleaned from this article I tried to install a USB modem, but, after much tweaking, I still haven't been able to connect to the modem. It is surprising that you gave so much attention to USB cameras, scanners, etc., but left out the ubiquitous modem. Could you please give a quick list of commands and steps needed for

installing a USB modem under Linux 2.4.2?

The modem is a Jensen v90 56k external modem, which works fine under WIN98 (sorry) installed on the same machine.

Terje V. Sorensen

The big problem with many USB modems is that, because USB is a bus protocol, rather than a serial protocol, they can get away with making cheap software driven modems, rather than selling proper hardware modems. If your modem is a hardware modem, which you can check in Windows, then you can use the standard serial modules from the USB system, and connect to it like a regular modem, through */dev/ttyS2*, or whatever it is allocated. If it's a software modem, you may like to check out www.linux-usb.org, or www.linmodems.org, but do not be surprised if you're out of luck in that department.

Ghostscript

I have two identical Gateway PCs, one in my office and one at home. I run Win98 on one disk and Red Hat 7.2 (recently upgraded from 7.0) on the second drive on each system.

On the office system, I installed the printer driver for my LaserJet-6P. It took three minutes and works fine. On my home PC, I have tried to install a printer driver for my Xerox-M750. Whatever I do, and whatever I follow, all that happens is that the green light blinks and nothing comes out. I also tried an old Epson Stylus 800 but this printed two lines followed by a page throw. The printing was fine, but the page throw problem was irksome, to say the least. Both printers work fine with Windows98, so presumably, it's not the printers, the cable or the connector.

The linux printer databases claims that the drivers for both

printers work OK. I purchased the Deluxe edition from Redhat, but their 30-day on-line support simply informed me that printing in Redhat is public domain, so they were unable (or unprepared) to help.

I'm a Linux advocate and have been developing real-time software (including operating systems) since 1970. I have hunted around the web for linux printing problems and guidelines, and I seem to have followed the instructions. However, it's this sort of problem that will make people give up with Linux and revert to MS. It's not a lot to expect installation of a printer driver for a dot matrix printer on a parallel port to be straightforward, is it?

Any help would be appreciated.
Dave Allerton

The Xerox M750 printer uses the HP PCL3 language, which is not standard in Ghostscript distributions. Ghostscript is the package *lpd* uses to interpret



Kswapd follow up

First off all THANK YOU ! you posted 2 of my letters in LFX23. The first was IPchains and the advice really helped and improved my system.

The second was on my Packard Bell Spirit 500 hanging on kswapd when I upgrade from redhat 7.0 to 7.1 or 7.2

After much research I am sorry to say that your advice (good as it maybe) did not apply to my situation. To test my theories I

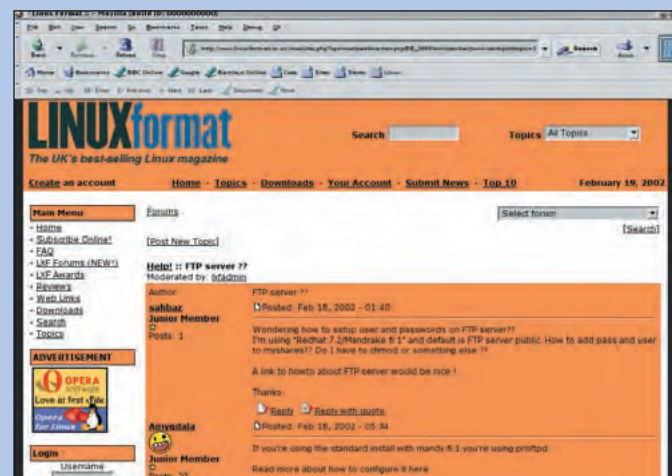
followed your advice and tried various large swap partition settings, all of these failed, I then tried your other advice or upgrading individual packages to Red Hat 7.1 levels. This worked until I recompiled the kernel. I moved from a 2.2.16 kernel to a 2.4 kernel – all went well until the reboot. After the reboot kernel hung on kswapd. I booted from my rescue floppy (forward thinking) and recompiled back to

2.2.16 which worked fine. I then moved through 2.2.18 and 2.2.19 – all working, as soon as I hit 2.4.x kswapd hangs my system. I have searched through Red Hat's bugzilla, I found this and added to it bug number 39089. It appears other users of the Packard Bell spirit range and a few other makes are having problems with the 2.4 kernel (or Red Hat >7.1). I would be interested in hearing your views on this bug and possible causes as Red Hat don't seem to be doing much.

I would also be grateful if you could publish this letter with my email address, so that other users having the same problem could let me know so that I can try to research a fix of sorts.

Matt Darcy,
matthew@darcy.demon.co.uk

The VM sub-system has changed drastically between 2.2 and 2.4, so it's not surprising that some machines are having problems with the new code. Maybe other users of Packard Bell hardware with Linux who have had similar problems could share their experiences on the *Linux Format* forums.



"It's good to talk". The *Linux Format* forums are an ideal place for readers to discuss their problems and maybe find another poor soul who is in the same position.



The trusty tiger is the ideal way to test your printer if you're not sure what isn't working properly.

« PostScript input and output whatever your printer likes. If Red Hat 7.2 does not have PCL3 support in *Ghostscript*, then you need to download it from <http://home.t-online.de/home/Martin.Lottermoser/pcl3.html>

If it is not printing anything at all, then it may be that *lpd* is not happy with what is being executed, and not sending anything at all to the printer. The quickest way to check is to try printing something manually, with;

```
$ gs -sDEVICE=pcl3
-sOutputFile=/dev/lp0 tiger.ps
```

If the *Postscript* document prints correctly, then it is certainly a problem with *lpd*, or the config, so it may take a little digging around in log files in order figure out what the problem is.

You're quite right that this should be a basic thing for Linux to do, and indeed, printing is a very easy thing to get going under many distributions, but it would appear that RedHat did not quite get it right when they distributed 7.2.

Yet more USB

Q I've got a problem which is seriously limiting the uses of my Linux installation. The computer is a dual processor Pentium III, on an Epox EP-D3VA motherboard with the VIA Apollo Pro133A chipset. I've installed SuSE 7.1, dual boot with Windows 98. I've got most things under control but USB system just doesn't work.

Kernel messages recorded when the USB host controller module (*usb-uhci.o*) is loaded manually with the printer switched on are:

```
Jan 20 21:49:24 server kernel:
usb.c: USB new device connect,
assigned device number 2
Jan 20 21:49:27 server kernel:
usb_control/bulk_msg: timeout
Jan 20 21:49:27 server kernel:
usb.c: USB device not accepting
new address (error=-110)
Jan 20 21:49:27 server kernel:
usb.c: USB new device connect,
assigned device number -1
Jan 20 21:49:30 server kernel:
usb_control/bulk_msg: timeout
Jan 20 21:49:30 server kernel:
usb.c: USB device not accepting
new address (error=-110)
Jan 20 21:49:30 server kernel:
usb.c: USB disconnect on device -1.
```

I don't know how to interpret these messages so I have no idea how to proceed. What do think's going on?

Richard Russell

A SuSE 7.1 is a very old distribution, in terms of kernel development, and since you have a machine with a particularly new chip set, it may be best to upgrade to a new version of SuSE before messing around with USB too much. The other option may be that you're using the wrong UHCI module for USB, as there are two, so you

might want to **modprobe uhci** and see if that helps any. As the USB subsystem detects the printer, then it must be working to some extent, although without upgrading the required kernel modules, it's somewhat difficult to decide if it's the kernel at fault, or a real problem with the USB hardware which you have.

Programming

Q I would be grateful if you could help me with a decision I need to make about learning to program.

I have recently installed Mandrake 8.0 on my home computer and have decided to learn a programming language but with such a large amount of languages available I find it very difficult to decide which one to learn. I have experience of Pascal, from my college HNC, which I found easy to pick up, but have found C and C++ quite difficult to learn in comparison.

I am looking for a language that is as easy to learn, yet powerful, as I would eventually like to contribute to the Open Source community if possible. At present I am reading about Python and Perl, that claim to be very good, but I have no experience of using these and would be grateful for your advice.

Phil Jenkins

A This is one of the hardest questions to answer because it really depends what's going to suit you best and what sort of programming you plan to do.

The first thing I would say is that, even though you may find it hard, it is worth persevering with C. The vast majority of open source software is written in C or C++, including the kernel and all of the system tools. Learning C will not mean you are able to create your own programs, but also contribute to the majority of other Linux projects. However, Python is a very powerful and easy to learn language. It's talents mainly lie in scripting, but you can develop GUI applications with it too. Perl is also very popular, though somewhat harder to master.

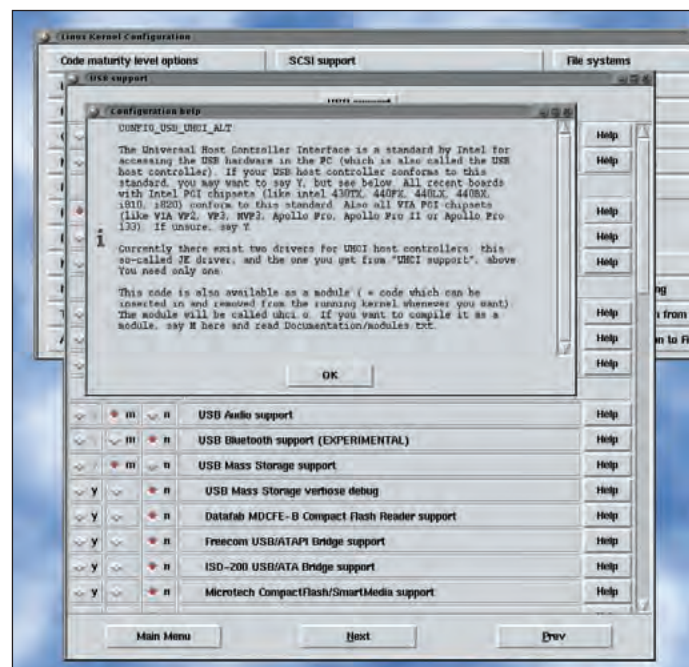
DVD DRIVE

Q I got your magazine (Dec 2001) with the DVD. I have a laptop with a DVD drive and Red Hat Linux 7.2 installed.

HOW THE HECK DO I MOUNT THE DVD ON MY FILESYSTEM?

I suppose you have a handout for this somewhere. I tried installing the UDF kernel loadable module and then mounting with `mount -t udf /dev/hdc /mymount` but it complained (even though *udf* now appears in */proc/filesystems* and */proc/modules*).

Tom Toffoli



The Linux USB sub-system has two modules for UHCI based chip sets, so it takes a little experimentation to decide which one is best.

A Due to the lack of out of the box support for UDF, and there not being a particular need for it for Linux DVDs, our DVDs still use the ISO9660 filesystem, so in effect they are just big CDs. Linux support for ISO9660 isn't restricted to a particular size, so this works pretty well. As most distributions just see DVD drives as identical to CD ROM drives, this works well for everybody. If you can read normal CDs in your DVD drive okay, there should be no problem with our DVDs.

```
mount -t iso9660 /dev/dvd
/mnt/dvd -o defaults,user,ro
```

should do the job for you – obviously substituting your own devicename and mount path, and including any particular mount options you like.

No joystick joy

Q Wonder if you have had any experience of installing an analogue joystick on the SuSE Linux 7.3 box?

I contacted the author of the joystick drivers and he has stated that SuSE's 7.3 kernel has a bug which prevents a joystick installing. I have tried getting info from SuSE, but they say they cannot help under the 90 day support scheme. This I find most frustrating if the kernel is at fault.

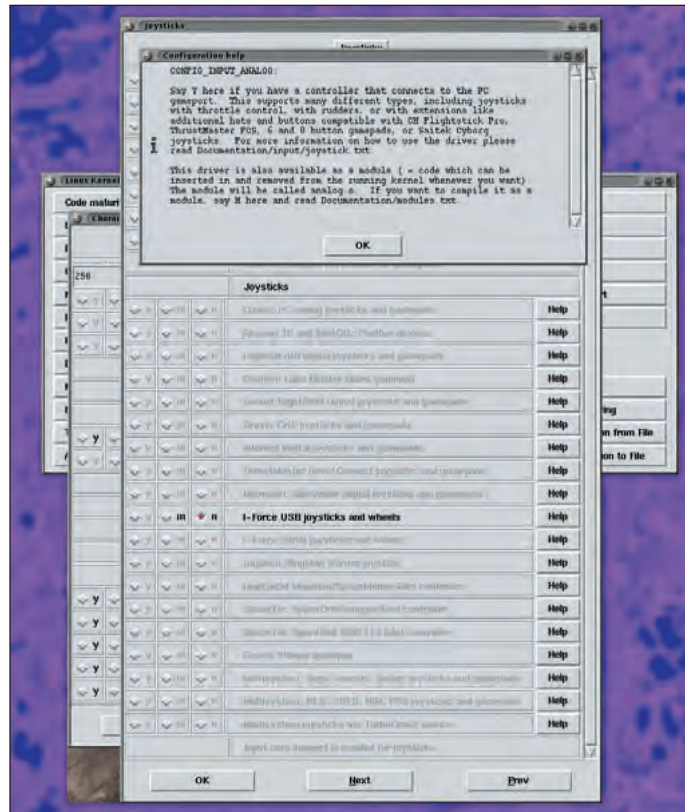
Over the past two years I have managed to install a CD writer, a dvd player (*Ogle*), an Epson Photo 1200 printer, an Epson scsi scanner, Internet connection etc. Why can't I have an operating joystick, so that I can run *Flight Gear*? Please help, you are my last hope!

Michael Griffiths

A If the author of the joystick drivers in the kernel suggests that it is non-functional in SuSE 7.3, then there is probably little you can do, other than rebuild the kernel, using a standard kernel distribution, or indeed, one from SuSE's FTP site. Even if SuSE can't help you, it is quite likely that they have released new kernel builds for people running 7.3 which offer both the regular kernel upgrades, along with a functional joystick driver.

IDE weirdness

Q OK, here's a weird one. I've asked sysadmins at work, trawled the newsgroups and



Setting up a standard analog (sic) joystick under Linux should be no problem, but SuSE don't seem to have got it quite right.

posted stuff as well – but all to no avail. My problem is this...

I have an old(ish) Pentium box (circa. 1996), that I installed Red Hat 7.0 on, which I *telnet* to from my main machine (saves dual-booting from windows). I do this using a crossover cable between two Realtek 8139 NIC's. Because the original hard drive was rather small, I installed a larger drive. The actual specification of the new drive is UDMA-133 7200rpm, which I realise will not be working at this level, but also realise that it should be perfectly backwards compatible.

Recently I decided to upgrade to Red Hat 7.2, which I did and the installation went smoothly. I was able to *telnet* to it fine, as well. However, after about 5 minutes – just after Red Hat runs those *cron* jobs that make your hard drive whirr away for several minutes – I hear an audible "click" sound from the hard drive and the box "dies". I can no longer *telnet* to it and I can no longer even ping the box. it is dead to the world, and to revive it you have to re-boot.

Strangely, though, there is one other option that seems to bring the box back to life. If I plug either

a keyboard or a mouse into their respective PS/2 ports after the box has "died", then suddenly the hard drive whirrs back into life and I am able to *telnet* to the box again. *N.B.* none of this used to happen with Red Hat 7.0 – all used to be well with that version of the distro.

I am thinking that maybe the older motherboard architecture is having problems with the new drive – but why did it not fail with Red Hat 7.0?

I am also thinking that maybe the new 2.4 driver for the Realtek 8139 (*8139too*) is giving me grief – but it would seem unlikely considering the symptoms described above.

I cannot get my head round it, and am hoping neither to have to roll-back to 7.0, nor upgrade any more hardware! Any ideas to what on earth may be going on would be greatly appreciated!

Andrew Hall

A The main difference between 7.0 and 7.2 is that 7.0 used Linux 2.2, and 7.2 uses 2.4 as default. 2.4 has upgraded IDE drivers, which are probably causing the problems you are seeing. It also depends upon what the specific

kernel options were when RedHat built their 2.4 kernel distribution, as it could be down to DMA being enabled as default, or a conflict with a specific IDE feature which causes your machine to lock up. It would be interesting to know if it's high-I/O of the drive which causes the machine to lock-up, or if it's when the machine has a high CPU load, or even when it tries to access a specific location of the drive.

A quick solution may be to build a 2.2 kernel, or download a Red Hat RPM of a 2.2 kernel, and use that instead of 2.4. If that works correctly, then you may want to build yourself a 2.4 kernel from scratch, using the source code from ftp.kernel.org, rather than a pre-compiled Red Hat version, then see what happens if you only compile in the bare minimum of features. Quite why plugging in a keyboard or mouse fixes the problem is intriguing and, as you said, it's unlikely that the NIC driver is causing the problem which you are seeing with the hard drive. One must also wonder what exactly is causing the 'click' of the hard disk, and it may be that the 2.4 IDE drivers are not controlling the hard drive correctly. **LXF**

Submission advice

We are happy to answer all sorts of Linux related questions. If we don't know the answer, we'll find out for you! But in order to give you the best service, it helps a lot if you read the following submission advice.

- Please be sure to include any relevant details of your system. "I can't get X to work" doesn't really mean anything to us if we don't know things like what version of X you are trying to run, what hardware you are running on.
- Be specific about your problem. Things like 'it doesn't work' or 'I get an error' aren't all that helpful. In what way does something not work? What were you expecting to happen? What does the error message actually say?
- Please remember that the people who write this magazine are NOT the authors or developers of Linux, any particular package or distro. Sometimes the people responsible for software have more information available on websites etc. Try reading the documentation!

We will try and answer all questions. If we don't answer yours specifically, you'll probably find we've answered one just like it. We can't really give personal replies to all your questions.

WRITE TO US AT:
Linux Format, Future Publishing, 30
Monmouth Street, Bath BA1 2BW or
email: lxformat@futurenet.co.uk

Reviews >>>

All the latest software and hardware reviewed and rated by our experts

LXF verdict explained

Each review is accompanied by a Linux Format Verdict to help you to assess the product at a glance (it's no substitute for actually reading the review, though). We award scores out of ten in the following categories:

Features: Does it provide the functions you need? Is it innovative?

Performance: How well does it do its job? Is it fast and reliable?

Ease-of-use: Is the interface well designed? Is the documentation well written, helpful?

Value for money: Does it have a competitive price?

For those who like numbers, the Linux Format Rating is a score out of 10 summing up the overall excellence of a product. It will usually, but need not be, an average of the above categories. We award scores as follows:

10 The close to perfect product.

8-9 Good, but has a few niggles.

6-7 Does the job, but needs work.

5-4 Average.

1-3 An utter disaster. Back to the drawing board.

The Top Stuff Award

If we really, really like something — we really think that a particular piece of software, hardware or any other sort of ware is the best stuff around — then we'll give it our Top Stuff Award. Only the very best will be chosen. It's not guaranteed to all products that score highly.



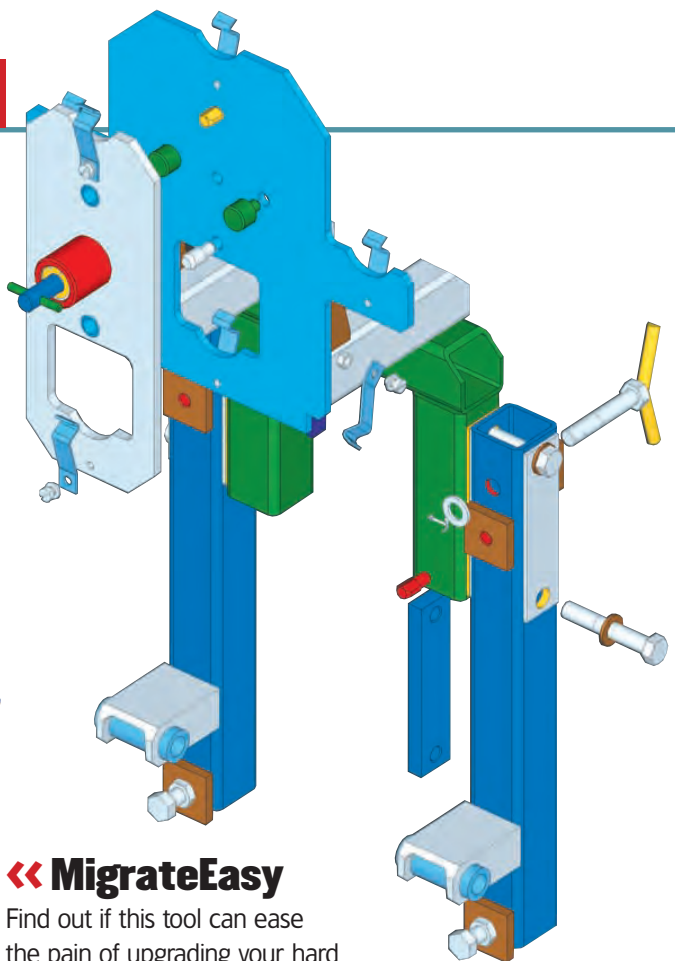
THIS MONTH...

VariCAD >>

2D and 3D design made easy with this competent engineering CAD package. What quality of package can be found beneath the quirky interface? **p28**

Sorcerer Linux

Ultimate control of your installation in this DIY distro which takes all the latest source code and builds it to your spec **p31**



<< MigrateEasy

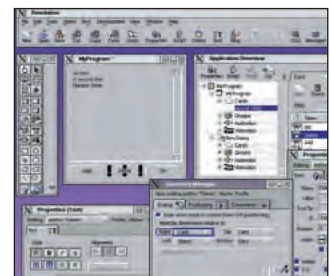
Find out if this tool can ease the pain of upgrading your hard disk. Is its convenience competitive enough? **p32**

Omis

Well established RAD suite for multi-tier application development. Does the latest point release match earlier promise? **p33**

Revolution >>

The return of Apple's eighties RAD, HyperCard, updated for a cross-platform world. Can it compete with Omnis Studio and Kylix 2? **p34**



<< Books

Linux Cookbook; Zope content management; Linux in the enterprise space and eXtreme Programming with Java this month **p36**

ENGINEERING CAD

VariCAD 8.0-0.1

Nick Veitch goes back to the drawing board with the latest release of this popular CAD package.

Professional level CAD software for Mechanical Engineering tasks. DWG/DXF/IGS file compatibility. You may also want to check out VARKON.

■ **DEVELOPER** VariCAD

■ **WEB** www.varicad.com

■ **PRICE** \$399 (download) + \$99 for box/manual/CDs

Although there are plenty of 3D related apps for Linux, good CAD packages are rather thin on the ground. One example that stands out is *VariCAD*, with around 10 years of development, and plenty of top-flight users. A Linux version of the software has been around for some time, but this latest release brings us up to version 8.0-0.1.

There are many types of CAD packages, dealing with many types of design. *VariCAD* makes no pretense that it is intended for anything other than engineering design. If you want to use it for other purposes (architecture, electronics, horticulture or whatever) you may find it is possible to some degree, but all of the defined objects, tools and layouts are focused towards mechanical engineering. In many ways this doesn't matter so much for 2D drafting – a line is after all, still a line whether it represents part of a push-rod, wall or plant pot – so it's worth considering for other jobs.

Install and configure

VariCAD is supplied as RPMs only, for Mandrake, SuSE and Red Hat (and for

only the latest versions of these distros – see the *Requirements* box), although the Red Hat RPM may well install on your distro if it is Red Hat-based. If you don't have GLX enabled in *Xfree86*, or fail the hardware or software

requirements for OpenGL support, the software will still run, but obviously without the OpenGL modes available.

No menu item is created for *VariCAD*, though of course you can add one yourself. Simply entering **varicad** in a console will start up the software, though this may be where you may come across our first niggle.

VariCAD seems to expect to run from your home directory, where the installation process has created a sub-directory to store your *VariCAD* docs. When you run the software from anywhere else you get error messages about the lack

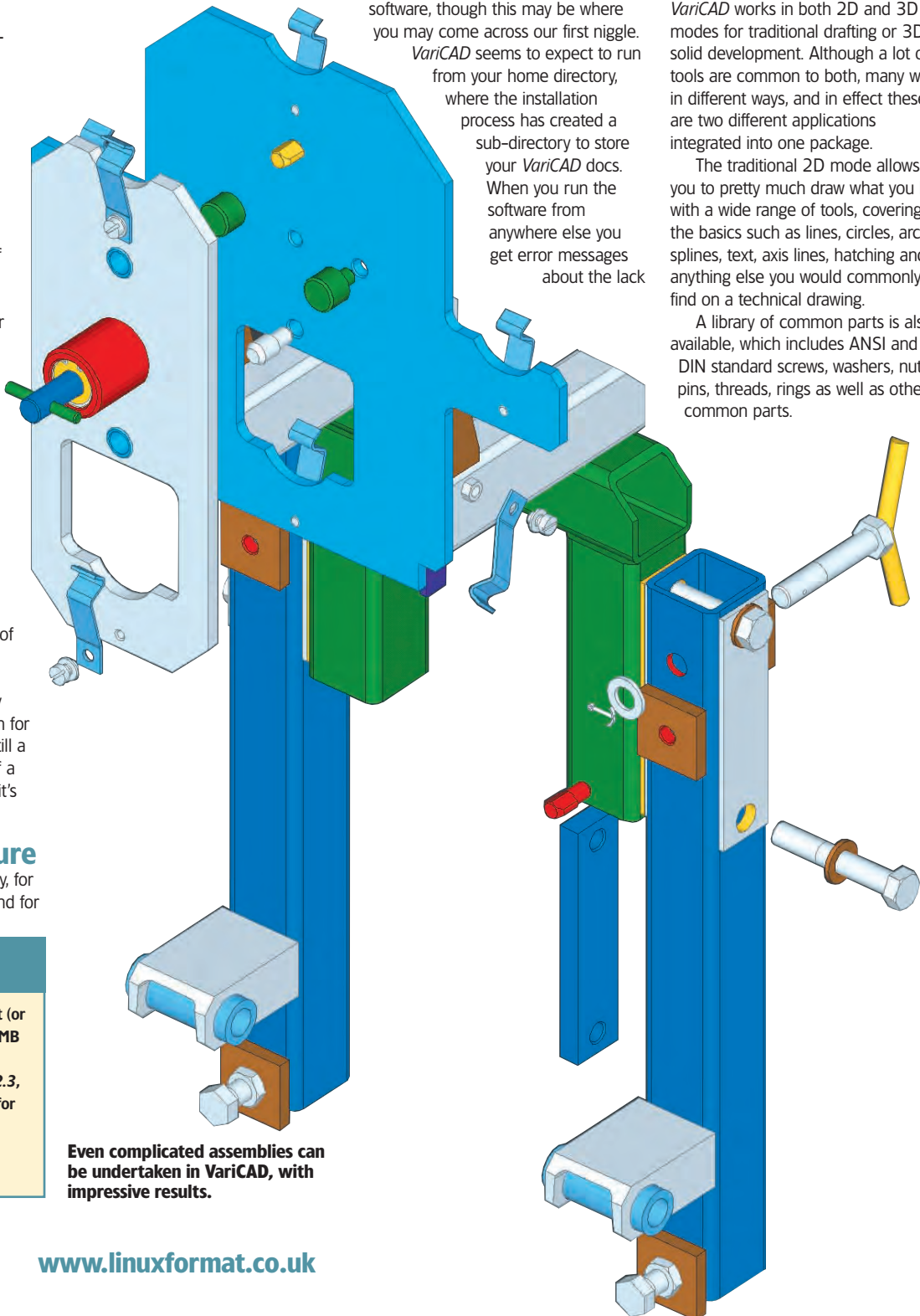
of a draw folder. You can cancel these and navigate to the folder that already exists when loading and saving, but surely this sort of info should have been stored in a config file?

Modes

VariCAD works in both 2D and 3D modes for traditional drafting or 3D solid development. Although a lot of tools are common to both, many work in different ways, and in effect these are two different applications integrated into one package.

The traditional 2D mode allows you to pretty much draw what you like, with a wide range of tools, covering the basics such as lines, circles, arcs, splines, text, axis lines, hatching and anything else you would commonly find on a technical drawing.

A library of common parts is also available, which includes ANSI and DIN standard screws, washers, nuts, pins, threads, rings as well as other common parts.



Even complicated assemblies can be undertaken in VariCAD, with impressive results.

Requirements

Hardware: PentiumII or equivalent (or better) processor, 64MB RAM, 30MB disk space

Software: *Xfree86 4.1* or later, *Qt2.3*, *KDE libraries 2.2.x*, *OpenGL/Mesa* for OpenGL support. RPMs only are supplied for Mandrake(8.1), Red Hat(7.2) and SuSE(7.3)

LinuxFormatReviewsVariCAD

« visualise how the components will fit together – thankfully the free rotation of the view window helps greatly with this. The view can be rotated, scaled and panned at any time, even in the middle of solid generation or positioning operations, so you can get a clear view of what is going on. Up to eight specific viewpoints can also be saved and recalled by pressing on the appropriate toolbar icon – a real timesaver.

Output of your finished work may be a little different. For 2D drawings there is support for printing and plotting. It may take some time to perfect the settings here if you are using a printer rather than an HPGL plotting device, because lines can be either too chunky, or too thin to see.

For 3D work, you can't print out the image, or even save a rendered view to disk, which is a little restrictive. Yes, you can export the objects and use a standalone render system, but it would have been nice to have even basic image output functions.

In use

Undo, for some reason, doesn't work on construction lines. There doesn't seem to be any real reason for this, it's just a flaw in the software. It isn't too difficult to delete construction lines, but it's a little harder to recreate ones you have deleted by mistake.

Little issues with the ease of use of the program begin to add up when it comes to more complicated operations like the creation of solids, or copying and translating solids. It took me a good half an hour to work out how to duplicate a simple support bar object I had created – it would have been far simpler to recreate an identical object than try to copy it. It is possible, but the documentation is a little vague on the exact steps necessary – it isn't as simple as say, choosing a 'Copy' function, selecting the object and selecting the position in which to place the copy. It involves using the translation interface and selecting the right buttons in the right

Display

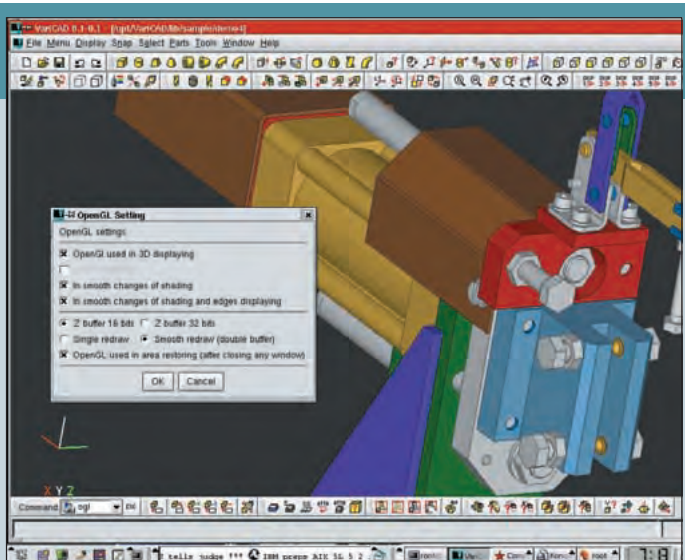
All round viewing

Controlling the view window is very easy and very fast. In 2D mode, the Shift and Control keys can be used to zoom and pan respectively. The redraw is pretty fast, enabling you to move around the drawing quickly.

The 3D display makes use of OpenGL to render the objects, so certainly make sure you have Mesa and the appropriate libraries installed for your card. Holding down both Shift and CTRL allows you to rotate the 3D view smoothly, which is a great boon for seeing in intricate detail, or for seeking the exact angle to zoom in on.

Various OpenGL functions can be set to refine the display, including whether to highlight the edges of faces or smooth them, which helps customise your view for different purposes.

If you don't have an OpenGL capable card, you can still design in 3D.



OpenGL makes the 3D mode work well – if you don't have support, this mode will be slow and somewhat more confusing.

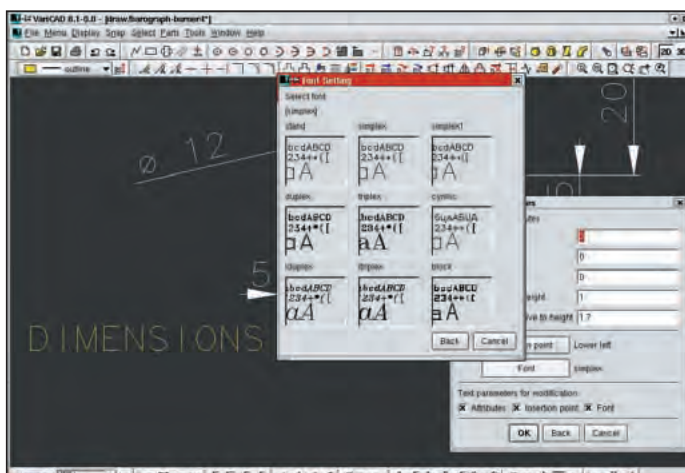
The display won't be so nicely shaded, so fast or display in 3D while being moved, but it works well enough to

see what's going on, and, assuming you have a 500MHz+ processor, should be fast enough to be useable.

sequence from this pop up panel.

The toolbars work well, and thankfully the banks of similar buttons can be moved around the fixed panel, taken off or used in floating windows. This means it is reasonably easy to configure your workspace the way you want it. The use of the KDE libs not only supplies some system support, but also makes the interface pretty straightforward to use on the whole.

Most of the tools are intuitive (apart from those mentioned here) and the extensive use of tooltips on the toolbar icons will soon get you going. If you are stuck, there is a manual of sorts. You can buy the printed manual for the software, but if you go for the download option you'll get it in HTML format. It is well structured and fairly easy to find the relevant parts of interest, but some parts are less complete than others. Complicated procedures are often explained with a few lines of text.



Dimensioning and text labels are easy – there is even a choice of fonts, though they may all look somewhat similar.

Sometimes this is simply not enough to understand what is going on, and some trial and error is called for.

Conclusion

This is by no means a flawless piece of software, as you can see from some of our comments here. But don't get the idea that it isn't a competent and useful CAD tool – it is. At a substantially lower cost than AutoCAD, it may be ideal if you need a CAD system for engineering drawings. There are plenty of useful tools and features, that would in the long-term outweigh the niggles with the user interface.

The bits that work are excellent, and, with a few reservations, it should provide a useful tool for quite advanced design work. The 2D

aspects of the software seem better developed than the 3D counterparts, but whichever is your main point of interest, in the proprietary software world this is a bit of a bargain. [LXF](#)

File compatibility

Working with other CAD files

Obviously you may want to use objects or drawings created with other systems. This is useful because not only might you be upgrading from an older product, but your suppliers will often have drawings of components that you can import into your own designs.

AutoCAD seems to be the *de facto* standard for files, so VariCAD attempts to provide file compatibility with the

ubiquitous .dwg, .dxf and also .igs (IGES) file formats. There is no support for importing 3D models in any of these formats at the moment, though 3D models can be exported in IGES.

In practice, this system still has trouble with some files, even in 2D mode. If there is some particular data you need to be able to use, best check it out first by downloading the demo.

LINUX Format VERDICT

Ease of use	7/10
Features	9/10
Performance	9/10
Value for money	8/10

A very capable and competent CAD system, only let down by some user-interface niggles.

LINUX Format **RATING**
 **8/10**

HIGHLY CONFIGURABLE DISTRO

Sorcerer GNU/Linux

David Coulson reviews a distribution for those who complain about packages not being up to date.

Source distro without close competitors. Ideal for those who have to have a special setup.

- **DEVELOPERS** Lunar Penguin
- **WEB** <http://sorcerer.wox.org/>

There are so many distributions of Linux available on the Internet, it's often difficult to decide which warrant attention, and those which are simply modified copies of well known distributions, such as Red Hat and Debian. Most of these distributions disappear into obscurity, because they don't offer anything which does not already exist.

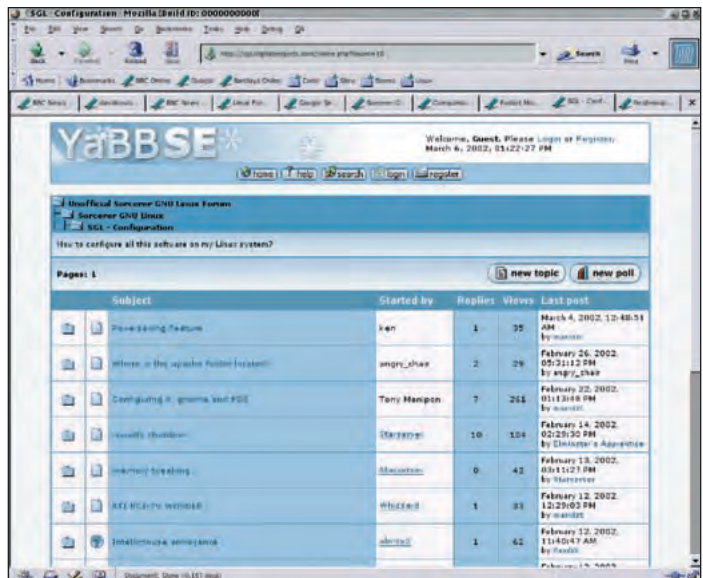
Sorcerer Linux is a very new distribution, but because it has no package distribution, then it really does not take much effort for the creators of Sorcerer to keep it up to date. Unlike most other distributions, Sorcerer GNU/Linux, or SGL, downloads the source packages from the official distribution FTP or HTTP sites, rather than requiring the distribution managers to create the packages when new revisions of commonly used programs or services are updated.

Sorcerer Linux is designed for system administrators and hobbyists who care to take a little extra time to setup their system properly, in exchange for having the most up to

date packages. The installer is not the most attractive or simple, and you have to use *fdisk* to create the partitions, for both storage and for swap. The up side of this is that it gives you extreme levels of control over the compilation and installation of the packages, and is a great way to learn about source code management and compilation if you've only ever stuck to using binary packages in the past.

The ISO distribution will install the basic packages to get your system installed, then you need to reboot it and use the Sorcerer management tools in order to install the packages you need. The standard packages are installed as binaries, so the first thing to do after installation is to rebuild the existing packages from source code, and reinstall them, which can take a while.

To manage your system, Sorcerer has a number of programs, but *sorcery* is the most important, as this is used to manage the packages which are already installed, and to reconfigure the optimisations used to build new packages. Much like *apt* in Debian, this can also download the package list from the Sorcerer distribution mirrors, so your system knows where to download everything from. Naturally, you need a fairly ample network connection, or plenty of time to wait, if you intend to keep Sorcerer up to date once you have



If you need support while installing or using Sorcerer, then there is an unofficial forum at <http://sgl.digitalprojects.com/>

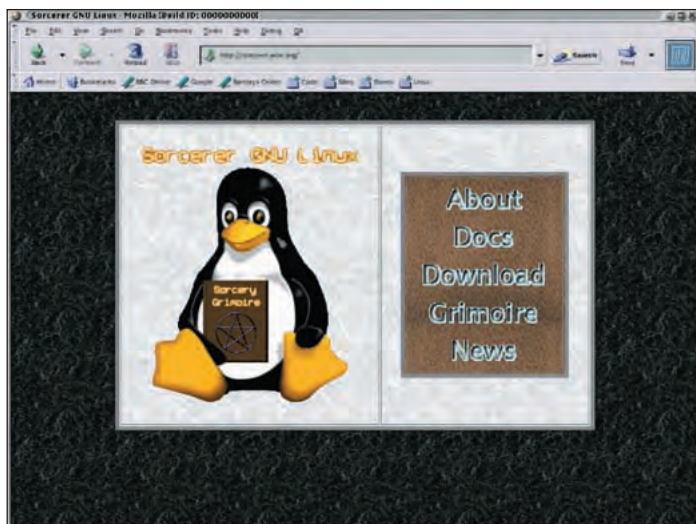
installed it from the ISO. You also need a very powerful machine to compile all of the source packages, particularly when it comes to memory availability, and indeed, the Sorcerer installer suggests at least 256MB of memory, plus 1GB of swap. If you ever do get to the stage where you need to swap 1GB of memory to disk, it's going to take a very long time to install the entire system.

Once installed, you have to pick out each individual package you want to install, although it does manage dependencies properly. It can take quite a while to pick out everything you want to install, let alone the time it takes to download and compile everything you want, so you can't do a completely non-interactive install – which can waste a great deal of time. Fortunately, once you have installed everything you need, keeping it up to date is very easy and requires little effort and time.

Installing new packages is easily done using the *cast* program, which will download the source code tarball, including all the dependencies it needs, then proceed to compile it with the previously configured optimisations and options.

The idea behind Sorcerer GNU/Linux is a very exciting, and possibly ambitious one, but Sorcerer does implement it very well. I

personally would not recommend it for anyone who wants to use their machine for production level service, instead a well maintained package based distribution such as Debian, would prove more stable and efficient. Many people have looked at Linux From Scratch, which is a source distribution of Linux which you compile by hand, and Sorcerer GNU/Linux could almost be compared to LFS as they both offer similar advantages over regular binary distributions. As far as support goes, don't expect a great deal of the official variety from the distributors of Sorcerer GNU/Linux, but there are mailing lists and web-based forums for discussion of problems with this distribution. **LXF**



While not being the most attractive web site ever created, sorcerer.wox.org has all the information you need in order to install Sorcerer GNU/Linux.

LINUX Format VERDICT

Ease of use	2/10
Features	6/10
Performance	7/10

Sorcerer GNU/Linux is a very unique and well specified distribution, although the hardware requirements and its bandwidth consumption will probably only go to convince people to use something else.

LINUX Format **RATING**
6/10

DISK MIGRATION UTILITY

MigrateEasy Deluxe

Deploying a new hard drive needn't be hard work, as **Jon Kent** finds out.

Disk mirroring utility that requires Windows to create the necessary boot floppy.

- **PUBLISHER** Acronis
- **WEB** www.acronis.com
- **UK DISTRIBUTOR** Interactive Ideas. Tel. 020 8805 1000
- **PRICE** £44.99

Although not a common task, at some point you will find that you need to upgrade your hard drive to cope with the extra demands put on your system. You are then faced with either starting again from scratch or moving some of your partitions to the new disk. Migrating a system partition to a new drive, however, is often more difficult than simply copying the files across, so some kind of mirroring utility is necessary. This is where *MigrateEasy* can help.

MigrateEasy aims to provide a simple method to transfer existing partitions to a new, possibly larger, disk. It supports multiple filesystems – including *ext2*, *ReiserFS*, *NTFS*, *FAT32* – and multiple operating systems – GNU/Linux, Windows, Solaris, FreeBSD and others. It can either be downloaded from the Acronis web site or purchased on CDROM. Unfortunately, to run *MigrateEasy* you do need to have Windows installed.

Installing *MigrateEasy* from Windows is straightforward. You don't actually run *MigrateEasy* from Windows, though. After the installation is completed, you are asked to create a boot floppy – which is used to start up your system and run *MigrateEasy*.

Pick a mode

MigrateEasy supports two modes of operation: automatic and manual. These do exactly what they say. The automatic method simply splits up the new disk into the same number of partitions as the old one, with the partitions proportionally the same size. This obviously is the simplest approach, but lacks flexibility as you cannot change the size of any of the partitions.

In manual mode you have several choices as to how you wish to transfer data from your old disk to the new

one. These are called 'As Is', 'Proportional' (the same as the automatic mode), and 'Manual'. Choosing 'As Is' copies the partitions from your old disk to the new one without changing partition sizes, and leaves the rest of the disk as unallocated. The 'Manual' choice allows you to size the partitions yourself and even allocate a partition that is not to be used, which is useful if you want to install an additional operating system, for example. You can also just partition a disk as additional storage, although you really do not need a tool such as this to achieve that. In both modes you are asked if you wish to make the new disk bootable, in which case *MigrateEasy* will copy across the MBR to the new disk.

Once you are happy with the partition layout you are presented with the 'Scenario Page', outlining how you have configured *MigrateEasy* to proceed. Up until this point *MigrateEasy* has made no changes to the disks: it is only once you press the 'proceed' button that the changes will be committed to disk.

We set up a test system with a 15GB disk and a 30GB disk and used *MigrateEasy* to transfer data from the former to the latter. The 15GB disk had a Windows 2000 partition and a Debian GNU/Linux partition of approximately equal sizes.

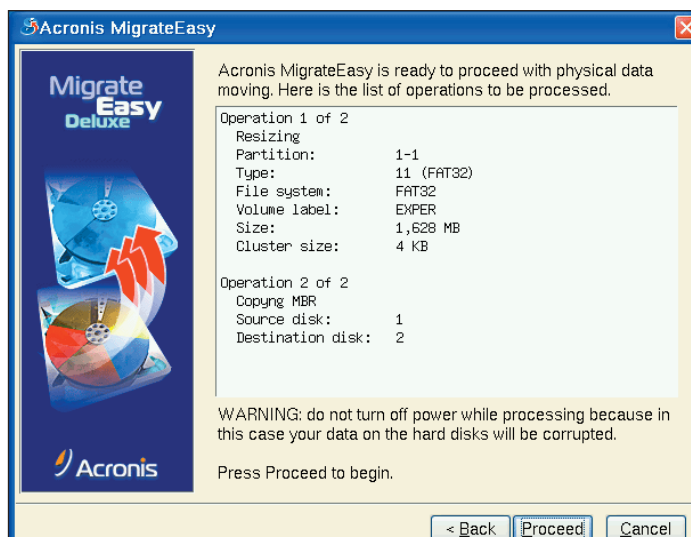
The automatic method was tested

first, as this is supposed to be the easiest approach. The transfer completed quickly and successfully, but when the system was rebooted *lilo* did not work, pausing at **LI** of the **LILO** prompt. This was not an unexpected problem – and is due to the way that *lilo* physically locates a kernel to boot – but the Acronis manual makes no mention of it. The fix is simple. Reboot the Linux system from a rescue disk and run *lilo* to rewrite the MBR.

For the manual method we

decided to set up four partitions, one of which was not going to be allocated. Again the transfer was completed quickly and again *lilo* would not work. Once this was fixed using the rescue disk Debian booted up perfectly. Windows 2000, however, was another matter. On booting it complained that there was no paging file and refused to allow any logins. Even running the repair utility did not seem to fix this. It looked like a complete reinstall would be required, so the manual method was only a partial success.

When it works, *MigrateEasy* is a useful utility to have. It was surprising that we had problems with Windows 2000 when using the manual method, since you would expect this to have been thoroughly tested against. Another issue is that you need to have Windows installed in order to set up the boot disk. However, you could use another PC which does have Windows installed to do this, so this is not a major cause for complaint. Migrating Linux partitions worked well using both approaches. **LXF**



MigrateEasy presents you with your choices for confirmation.

LINUX Format VERDICT

Ease of use	7/10
Features	9/10
Performance	8/10
Value for money	3/10

The number of operating systems and filesystems supported is impressive, but it is expensive for a tool that you will use infrequently.

LINUX Format RATING

6/10

CROSS-PLATFORM RAD

Omnis Studio 3.1

Richard Drummond take look at the latest version of this stalwart RAD suite for database and web development.

A package of visual tools for developing multi-tier applications.

- **DEVELOPER** RainingData
- **WEB** www.omnis.net
- **TEL** 01728 603011
- **PRICE** See box Pricing

Omnis Studio is a cross-platform RAD (Rapid Application Development) suite that can tackle any project from simple stand-alone programs to multi-tier, load-balanced, web-based applications. It provides an advanced visual environment to build and test such projects on Linux, Solaris, Windows and MacOS 9 and X. We last reviewed *Omnis Studio* in issue 10, and, although there are no major changes in version 3.1, it's a package well worth re-evaluating in the light of the latest developments in this area of the market.

Omnis basics

Omnis Studio in essence is very similar to *Revolution* (see review, pages 34–35) – although the scope of *Omnis* is much wider. It boosts developer productivity by enabling them to build database-driven applications from a range of pre-built components, and glue together those components with code written in a fourth-generation scripting language (or 4GL). The interface for your application can be a traditional GUI, a web-based interface or you can have both in a single project. The web-based interface can be served up as plain HTML or – with the *Omnis* browser plug-in – can make use of a similar range of GUI components as a stand-alone interface.

Database access

Built-in or remote: the choice is yours

One of *Omnis*'s key feature is its flexible support for database access via its custom Database Access Modules (or DAMs). The Standard Edition supports only local file-based databases, optionally via the built-in OmnisSQL engine. The

This plug-in is available for *Internet Explorer* and *Netscape*, and the latter plug-in works under *Mozilla* on Linux (we had no luck with *Konqueror*, though). The *Omnis* application server integrates with *Apache* via a supplied *Apache* loadable module.

The *Omnis* IDE is polished and easy-to-use. A comprehensive set of documentation is provided as PDFs (printed manuals are extra), and plenty of example and tutorial material is supplied. *Omnis*'s 4GL is not quite so high-level as some, but it is still easy to learn, and keeps actual coding to a



Pricing

Single-user development licenses:	
Standard Edition	£90.00
Enterprise Edition	£349.00
Web Edition	£799.00
Single-user runtime license	£60
Additional 16 users	£50

(All prices not inclusive of VAT)

minimum. *Omnis* Application Builder Wizard can even generate certain classes of simple application for you without requiring any code to be written at all. The integrated method editor simplifies entering the code that you do have to write.

Omnis's object model is a lot more flexible than *Revolution*'s and exhibits real object-orientation: objects have classes and you may declare your own sub-classes as you require. The IDE provides tools to navigate the inheritance hierarchy in your project, and the Interface editor lets you browse the properties and methods of an object.

The range of prebuilt classes included in *Omnis* is rich, including many of the components you would

expect in a general-purpose programming language.

XML?

Omnis Studio is a powerful and flexible visual suite, and the overhaul of its web-based application server with the release of version 3.0 catapulted it into a new league – although, it's reliance on a browser plug-in and its licensing model make it more suited to in-house and intranet deployment. My only real complaint with *Omnis Studio 3.1*, however, is one that remains from version 3.0: there are no tools to ease the deployment of applications. This becomes even more crucial with

server-side development, and *Omnis* needs to include support for, for example, the WebDAV protocol so that code can be automatically uploaded to speed up the deploy-and-test cycle.

The state of the market has moved on since *Omnis 3.0* was released. Web Services and SOAP are now deemed to be the next big thing, and every vendor of server-side tools is desperately trying to implement support. *Omnis* has none as yet, but a module for handling XML documents is currently in beta testing. Moreover, a new release of *Omnis Studio* itself is scheduled for release later this year.

As it stands, *Omnis* provides a solid – if not exactly cutting edge – platform for database-driven and web-based applications. If productivity counts for more than buzz phrases, why not give it a try. [LXF](#)

LINUX Format VERDICT

Ease of use	8/10
Features	8/10
Performance	9/10
Value for money	7/10

A solid suite of tools for multi-tier application development wrapped up in an easy-to-use IDE.

LINUX Format RATING

8/10

RAD TOOL

Revolution 1.1

Richard Drummond investigates a package where multimedia authoring meets cross-platform RAD.

Revival of Apple's radical '80s RAD tool, to compete with *Omnis Studio* and *Borland Kylix*.

DEVELOPER

Runtime Revolution Ltd

■ **WEB** www.runrev.com

■ **TEL** +44 (0)131 718 4333

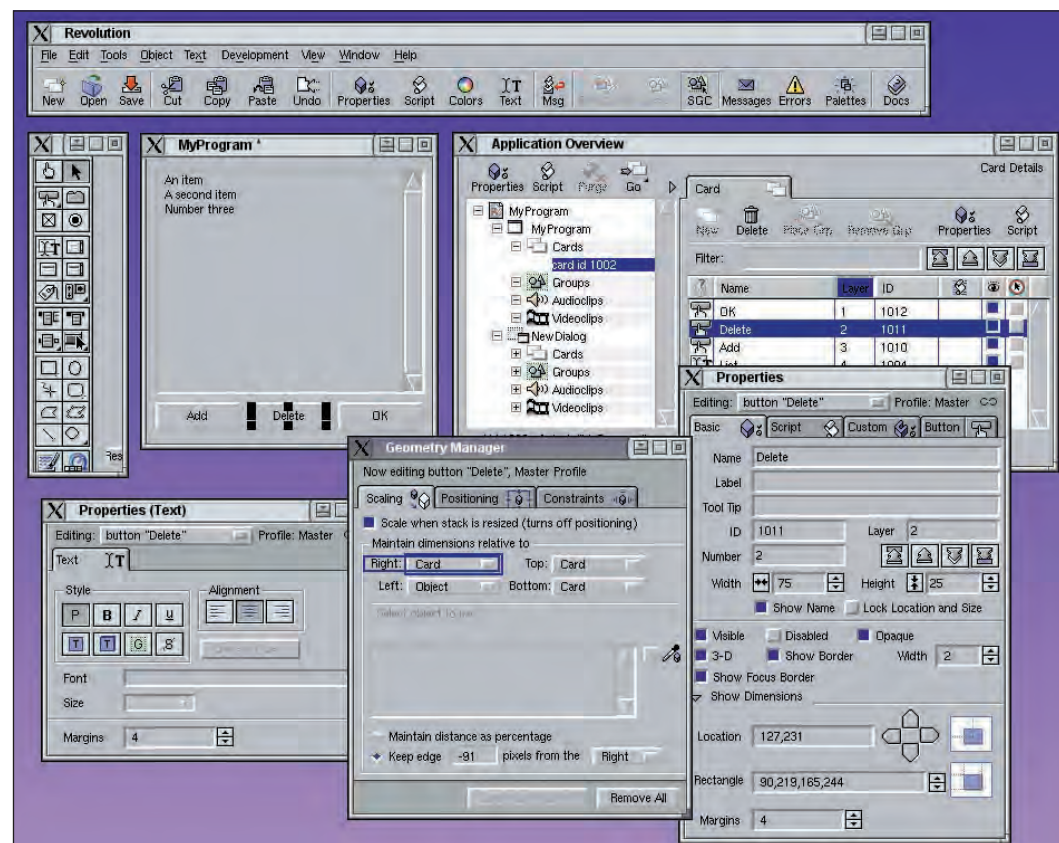
■ **PRICE** See box

Revolution is a cross-platform RAD tool aimed to be easy for non-programmers to use and to cut development times for developers. It lets the user build GUI-based applications from a wide range of prebuilt components with a minimum of actual coding. Revolution's programming language is a very high-level – almost English-like – scripting language called Transcript. Hence, no separate compile phase is required in Revolution. You can even edit the scripts of a running project and the changes will take effect.

You card

The *Revolution* programming model is based ultimately on Apple's *HyperCard* via *MetaCard* (see *Revolution evolution*). This is an object-based world, but the terminology may be different from what you are used to. Thus windows are stacks, panels are cards and widgets are just called objects. The stack is the fundamental building block, and an application can have multiple stacks and each stack can contain multiple cards – but only the top-most card is visible.

Revolution's IDE will be familiar to anyone who used a GUI builder before. You simply drag-and-drop components from the palette to your stack, modify any properties you require in the properties editor, and write any code necessary to glue these object together into a living, breathing application. The main differences in *Revolution* are down to its *HyperCard* heritage. *Revolution* provides a range of drawing tools with which you can paint in a card just like you would with a paint package; also, you don't need to worry particularly about grouping your components or using a layout manager. Hence, it's easy for beginners to get to grips with. If you need more finesse, *Revolution*



Revolution's IDE has itself been created with Revolution, and thus demonstrates the power of the system.

does provide a Geometry Manager tool which allows you to set constraints on the positioning of objects within a card and make their positions and sizes adapt as you require to window sizing. This makes it possible to create traditional types of GUI, but the implementation is rather belaboured.

The range of widgets supplied is very complete and extends on the set supplied by its progenitor, *MetaCard*. These have a switchable look-and-feel – either *Motif*, Windows or MacOS – no matter what the host platform is. Where *Revolution* excels over most IDEs, however, is with multimedia components. It is a simple matter to add images, GIF animations and *Quicktime* movies to your programs.

Although *Revolution* is object-based, it is not object-oriented. The only relationship between objects is that of containment: stacks contain cards contain objects. You cannot add extra levels to this hierarchy, and there's no real concept of class. You

Pricing

Starter Kit Edition

Free

Single-user Standard Edition

\$349

Single-user Professional Edition

\$995

Multi-user Professional Edition

\$3600 for 5 users

Educational licenses and cross-grades from MetaTalk available. See <http://www.runrev.com/revolution/info/moreinformation/licensing.html>

cannot declare your own classes of objects – except by creating them externally in C and using the plug-in API. Since there's no concept of an "is a" relationship, you cannot inherit a class to extend or adapt its behaviour.

Objects in *Revolution* send and receive messages. For example, a button widget is sent 'mouse down'

and 'mouse up' messages when it is clicked by the user. If this button has no handlers defined for these message, it will pass them up to its parent, the card object within which it lives. If no handler exists there, the card will pass it on up to the stack. The simplicity of *Revolution's* object model here show its advantages. Unhandled messages bubble up the object hierarchy, while properties are shared downwards (that is, an object inherits the properties of its parent). The user can choose to at which level it is most convenient to implement a handler.

Each object can have a Transcript script associated with it. Here you can write any event handlers or methods to handle custom properties of that object. Code is written in the script editor, and this provides a fast and error-free way of entering code, supporting all the usual syntax-highlighting and auto-completion tools. An online dictionary is included where you look up reference material on

Transcript's array of built-in functions. Transcript itself is very easy to learn and although its syntax seems overly verbose at first glance, because so much of the work is done for you by *Revolution's* objects, it often requires a lot less code to get the job done than traditional programming languages.

A promising feature of *Revolution* is its wizard for exporting your projects – either as a stand-alone binary for any of the supported platforms or as a collection of *Revolution* files. The latter would be most useful, since they are architecturally neutral, if some kind of player were provided to launch them – but there isn't one. Another problem is that MacOS executables for some reason can only be built when *Revolution* is hosted on the Mac.

We were unable to get the Distribution Builder wizard to build stand-alone executables under Linux at all – although it worked well under Windows and MacOS. This is one of the many cases where the Linux version of *Revolution* proved unreliable. Scores of run-time errors are generated by the IDE, seemingly at

random, and the system has problems cooperating with X window managers.

Aiming high

Revolution appears to be an attempt at retargeting MetaCard for more conventional development tasks. The IDE itself has been made over to look more like a modern RAD tool, and the extra libraries of tools added in *Revolution* – such as the Geometry Manager – are geared towards make traditional types of development easier.

One such area in which *Revolution* extends over *MetaCard* is with functions for database access. Runtime *Revolution* themselves admit that this, however, is very much a work-in-progress. Currently ODBC, MySQL and Oracle are supported, but only ODBC support is included in the *Starter Kit* and *Standard Edition*. The database functions are poorly documented at the moment and no real example code is provided.

The Runtime IDE provides a GUI-based Database Manager to simplify database programming, but this is woefully primitive. It provides a stack

Revolution evolution

Hypercard for the 21st century

Rapid Application Development tools might be fashionable now, but they are certainly not a new idea. Apple was one of the pioneers of RAD technology in the 1980s with its *HyperCard* system. *HyperCard* provides users with an integrated environment in which to build and test applications, or stacks, built out of prebuilt components glued together with scripts written in a very high-level language called HyperTalk. It might not be billed as RAD technology, but the game's the same.

HyperCard is for the Mac only, and rather limited in the types of applications you can build with it: *MetaCard*, an independent re-implementation of *HyperCard* launched in 1990, overcomes these limitations. It is cross-platform –

versions exist for most flavours of Unix and for Windows – and *MetaCard* stacks themselves are portable and will run on any platform with a *MetaCard* player. It's more general-purpose, too: *MetaCard's* scripting language – *MetaTalk* – extends HyperTalk with a library of functions equal to the power of those provided with many traditional programming languages.

Revolution is built-on the *MetaTalk* engine and is an attempt to bring *MetaCard* into the 21st century. It sports a more complete and more modern-looking set of GUI controls, a swankier IDE and adds various extra libraries of functions – such as the Geometry Manager and Database Manager.

and cards to allow you to graphically set-up a database connection and to implement and execute queries. This works, but is not an ideal solution: it is just so inelegant to encapsulate a database connection as a window. What they should really do is take a leaf out of *Kylix's* book and implement non-visual components.

These problems with the Database Manager highlight the limitations of *Revolution's* object model. While this model works well for simple projects, it doesn't scale well to larger and more complex projects. Of course it is possible to develop sophisticated applications with *Revolution* – the IDE itself is written in *Revolution* – but it will be hard work because the system doesn't have the tools to cope.

Revolution has a bit of an identity crisis. It is an excellent tool for knocking up quick one-off applications, doing multimedia presentation and prototyping GUIs – but it wants to be more than this.

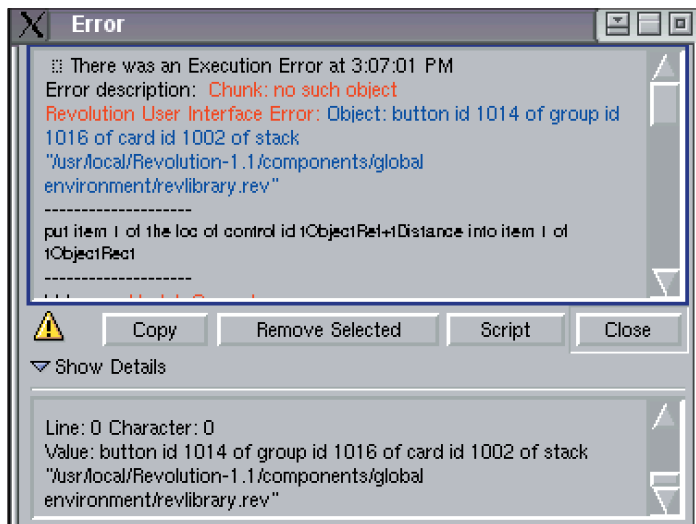
However, it is not in the same league as the big boys. *Omnis Studio* is perhaps its closest competitor, and *Omnis* is streets ahead in database connectivity and tools for building web-based applications. The pricing of *Revolution* moves it far away from its roots in *HyperCard*, but it just hasn't moved far enough to justify the price. Suddenly, *Kylix* seems a much better value investment. **LXF**

LINUX Format VERDICT

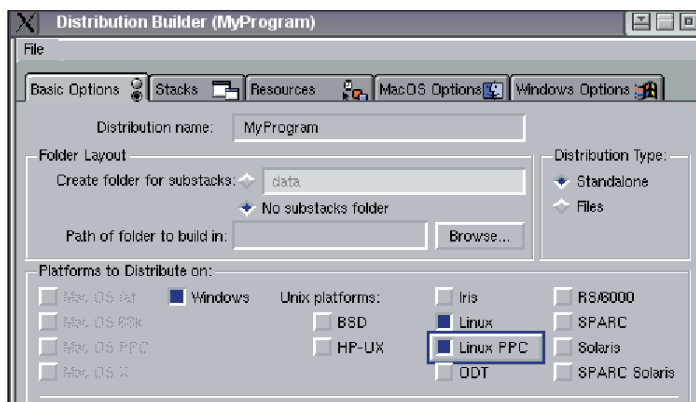
Ease of use	8/10
Features	7/10
Performance	5/10
Value for money	6/10

Shows potential, but needs more work if it is to shrug off its heritage and be accepted as a serious RAD tool.

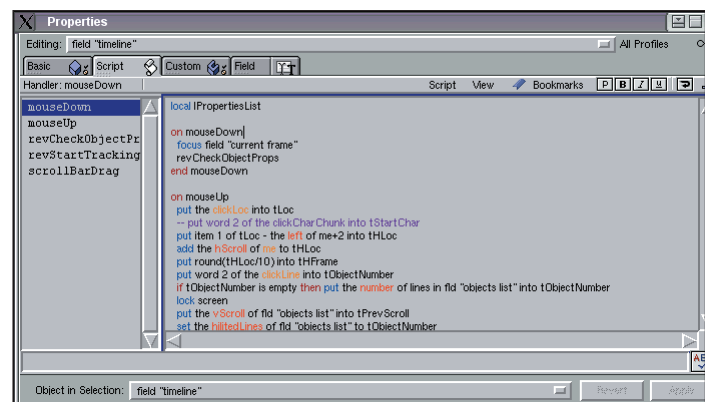
LINUX Format RATING
 **6/10**



You will see this dialog box frequently, because – on Linux at least – the *Revolution* IDE is quite buggy.



The distribution builder is potentially a great tool for deploying your projects, but alas it doesn't seem to work in the Linux version.



The script editor lets you write Transcript code to glue your application's components together.

The book of Zope

Nick Veitch discovers all he needed to know about Zope, in this well structured introduction.

- **AUTHORS** Casey Duncan, Beehive
- **PUBLISHER** No Starch Press
- **ISBN** 1-886411-57-3
- **PRICE** £25.95

Zope, as you probably know, is the python-powered content management /application development system from Digital Creations. While it provides a very powerful system for managing web sites and creating application, the fact that it uses new technologies and terminology has rather put some people off learning it. And that's exactly what this book tries to address. It begins with basic information about what Zope is, how to get it,

install and configure it. Subsequent chapters take the reader through the specific aspects of creating and managing a Zope site, with plenty of practical examples and screenshots (printed at a size where you can actually see what's going on).

The book is structured well, and pretty well written. A knowledge of HTML is expected, and an idea about Python is pretty much essential, but apart from that, all the concepts and ideas are explained well. If this is your first venture into the world of Object-Oriented work, some early chapters may be a little hard-going.

Chapters progressively cover administering Zope, dealing with DTML (the special Zope dynamic content format), security, users, classes



catalogues and databases – pretty much all the information you will need to construct a fairly complex site.

The principles of how to extend Zope using your own Python code is also touched on briefly, but as this isn't a Python scripters book, more info on this is available elsewhere.

The book is easy to navigate, and a reasonably thorough index should help you find the bits you need to know should you get lost. There is also an adequate glossary explaining terms you may be unfamiliar with.

I'm sure it would be possible to write a more detailed book on Zope. It may even be possible to write one that

is easier to follow, but as it stands this is probably the best Zope book for those who are actually interested in deploying websites, rather than over-indulge themselves in the technology behind the scenes. A good effort.

Linux Format VERDICT

A good all-round introduction and practical guide to using Zope.

LinuxFormat RATING

9/10

Open Source Enterprise Solutions

Jono Bacon dons his pin-striped suit to read a book on Open Source's growing rôle in enterprise.

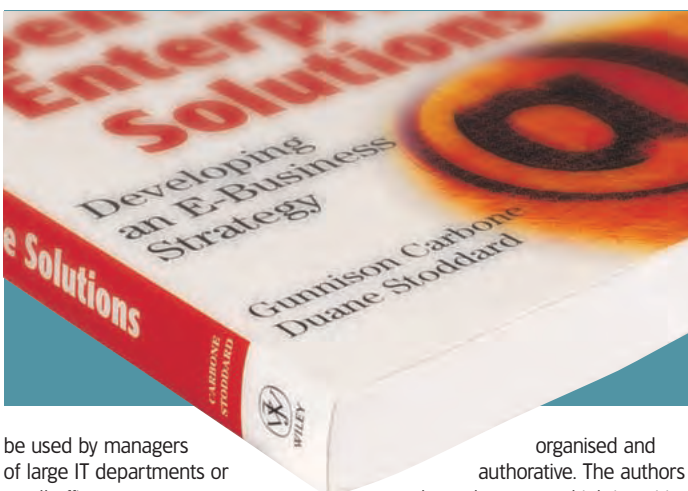
- **AUTHOR** Carbone & Stoddard
- **PUBLISHER** Wiley
- **ISBN** 0-471-41744-0
- **PRICE** £35.95

With the recent advances in usage of Linux within corporate environments and enterprises, Linux is no longer just a hobbyist's OS. This book aims to show how and where Linux can be used in this area.

Firstly, the book contains a wide array of topics concerning analysing and implementing Open Source in the enterprise. Topics include details on

the various Open Source technologies available, analysis of a strategy, integration of Open Source products and e-commerce. The sections on integration of a solution, data modelling and protocol implementation are of particular interest. The content is a mix of managerial, technical and analytical information; perhaps occasionally too technical for some IT managers.

The book covers networking, web serving, access to data stores and e-commerce, and communications. No specific requirement has been placed on the size of the enterprise that an IT solution should fit in, so the book can



be used by managers of large IT departments or small offices.

The only real disadvantage is that the book only really focuses on the use of Open Source in a server environment – desktop use is only briefly discussed. Granted, many people feel that desktop use of Linux is not so widespread, but nonetheless, some details on the issues surrounding using Linux on workstations would be useful. More information on interfacing an Open Source solution with a commercial solution such as Windows/Solaris/MacOS etc would also be useful.

Other than these small grumbles, the book is well written, clearly

organised and authoritative. The authors have done a good job in writing a clear and concise guide to implementing Open Source in a traditionally commercial environment. If you are looking to implement some kind of Open Source in your enterprise, I suggest you give it a look.

Linux Format VERDICT

Well written & comprehensive analysis of using Open Source in the enterprise space. Give it to your IT manager.

LinuxFormat RATING

8/10

Java Tools for eXtreme Programming

Richard Drummond learns how the XP mantra applies to Java coding.

- **AUTHORS** Richard Hightower and Nicholas Lesiecki
- **PUBLISHER** Wiley
- **ISBN** 0-471207-08-X
- **PRICE** £29.95

EXtreme Programming (XP) is a controversial new software engineering methodology that advocates automated and continual unit and integration testing. Java Tools for eXtreme Programming is a guide to practicing XP in enterprise Java projects using such open source tools as *Ant*, *JUnit* and *Cactus*.

The book is split into three sections. The first part covers the

principles of XP, gives an overview of J2EE deployment and introduces the examples used throughout the rest of the book and the third part is a reference section and lists APIs for *Ant*, *JUnit*, *Cactus*, et al.

The main body of the book, however, is the second section and this discusses the basic usage of the various tools covered and how to employ those tools to achieve automated building, testing and integration. The reader is led through building a project with *Ant*, unit testing with *JUnit*, in-container testing with *Cactus*, functional testing with *HttpUnit*, and so on. A case study J2EE application – a web-based pet store –

is used to illustrate each of these stages.

The aim of XP is pragmatic: it enables a team to deliver tested, integrated code at any point in the development cycle. Java Tools for eXtreme Programming adopts a similar get-the-job-done approach. The case studies serve as templates which can be used to apply XP principles in your own projects. This book does less well at explaining how these tools can be used more generally. A lot of space has been wasted which could have been more beneficially used to provide greater detail on the tools *Ant*, *JUnit* and so on. For example, the reference section

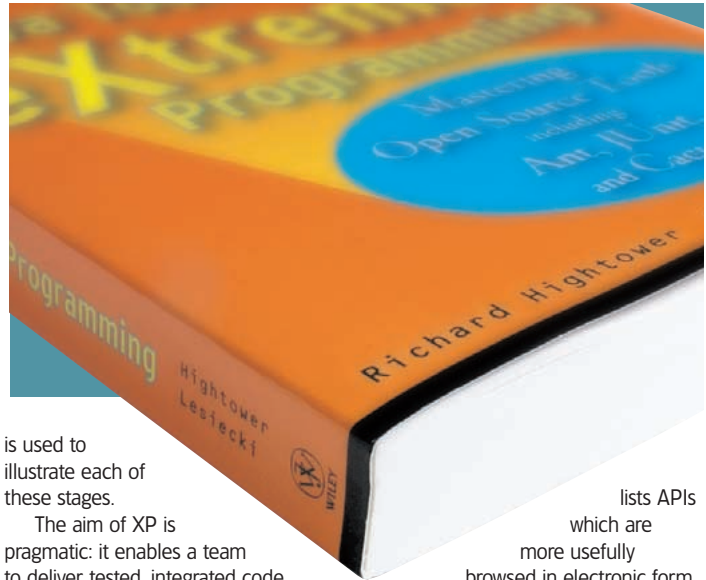
lists APIs which are more usefully browsed in electronic form, and too much space is devoted to listing source code which is also available on the companion website.

Linux Format VERDICT

A practical guide to applying XP principles in your Java projects, but doesn't adequately explore the tools.

LinuxFormat RATING

7/10



The Linux Cookbook

In life, as in the command line world, there is always something new to learn, says **Biagio Lucini**.

- **AUTHOR** Michael Stutz
- **PUBLISHER** No Starch Press
- **ISBN** 1-886411-48-4
- **PRICE** £23.99

The first time I saw this book with this funny cover featuring a penguin with a cook hat I thought that it must have been another reincarnation of the standard beginning-oriented tutorial on the basic of the command line. I have nothing against it, but after having used Unix and derivatives for several years, this type of books no longer attracts my attention.

My first impression could not have been more wrong. *The Linux Cookbook*

is indeed focused on the command line, and it is highly recommended to Desktop users that are afraid of typing commands in a terminal. However, it uses an approach different from many books I have seen so far: instead of listing analytically all the possible options of every single command (a procedure that often results in cryptic statements), it just focuses on single everyday tasks, and on what the author reckons be the best way to achieve them. The underlying philosophy is the old Unix idea that a given task requires just one small but very specialised tool. Here you see this idea at work. A similar approach results in an impressive extension of the ground covered. The simplicity and

the efficiency at the same time of the proposed solutions are astonishing. As a consequence, also the most experienced shell geek will have something to learn from this book. Graphical tools aren't forgotten either: programs like the *GIMP* get attention, but always in the spirit of accomplishing a given small task.

"Shortness" and "smallness" seem to be the key words of the book: the "recipes" hardly take more than a couple of lines and hardly more than five recipes fit one section. As a result, *The Linux Cookbook* has an impressive number of chapters, which range from the fundamental "What Every Linux User Knows" to more advanced topics

like text formatting, imaging, sounds and the internet, just to mention a few.

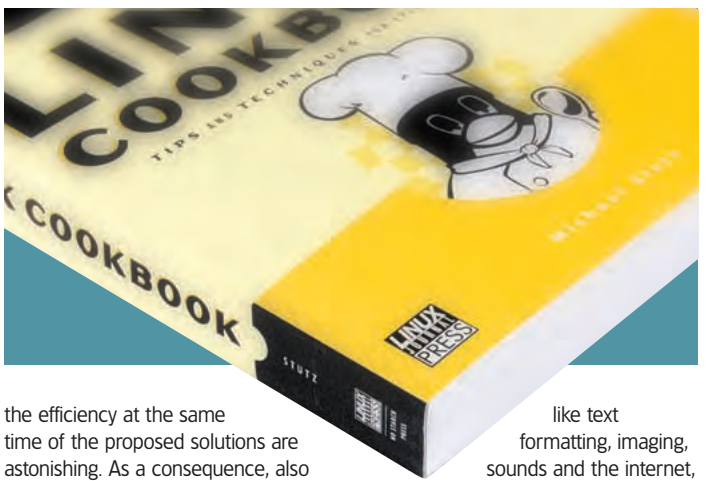
I have decided that this book deserves a place in my handiest shelf. As for you, don't take just my word for it: take a look yourself at <http://dls.org/cookbook>. **LXF**

Linux Format VERDICT

An impressive range of simple but efficient tips and techniques.

LinuxFormat RATING

9/10



Roundup >>

Every month we compare tons of software, so you don't have to!



DVD players

The bad old days of no legal DVD players are long gone. **Jon Kent** finds himself presented with an embarrassment of riches from the Free Software treasury.

Playing your bought-and-paid-for DVDs under GNU/Linux has not had the happiest of histories, what with the DeCSS debacle and subsequent legal battle in the States, which is outlined shortly. Luckily, there are several open source projects underway that allow you to play unencrypted DVD directly. As luck would have it these have all been designed so that you can play encrypted DVD (*i.e.*, the ones you bought) using a third party library or plugin which you need to download separately, which is very convenient. The legal issue with DeCSS is still with us, but is slowly getting clearer as time goes on.

This is probably a good time to provide a summary of how DeCSS came about and the legal bun fight that ensued after it was released. The most famous individual to be connected with the DeCSS code is Jon Johnsen, a sixteen year-old hacker from Norway. As a member of an international hacking group known as Masters of Reverse Engineering, or MoRE, Johnsen had been one of the individuals responsible for DeCSS, a piece of software designed to decrypt scrambled DVDs. The MoRE developers had deciphered and reverse-engineered the encryption method used on most commercial DVDs. Of course, once the Motion

Picture Association of America (MPAA) heard that their prized encryption method had been successfully cracked they were none too happy. The MPAA claimed that the hackers had effectively destroyed their ability to protect its content from unauthorized copying. Together with the DVD Copy Control Association, a consortium representing 400-plus DVD device manufacturers, the MPAA launched a trio of U.S. lawsuits charging DeCSS source code distributors (essentially the web sites that has the code posted for download) with copyright infringement and misappropriation of trade secrets. This started the legal injunctions

Our selection at a glance

- Xine
- VideoLan Client
- MPlayer
- Ogle

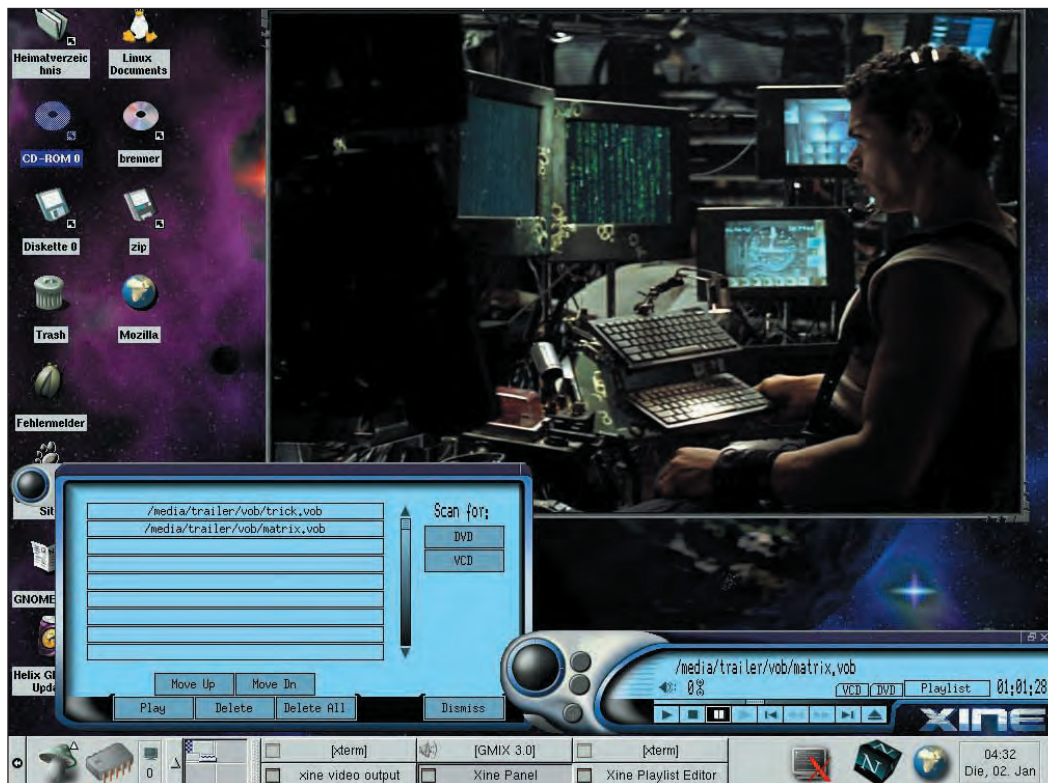
against any site that had the code available, either by publishing or linking to the DeCSS code.

Then on January 24 2000, the following message was posted on Slashdot by Jon Johansen :

"The National Authority for Investigation and Prosecution of Economic and Environmental Crime in Norway raided my home today and seized my Linux box, FreeBSD/Win2k box and Nokia cellphone. Not only I, but also my father has been indicted, since he owns the mmadb.no domain (webhotel) where my homepage(s) have been located. They also took me in for questioning which lasted 6-7 hours. It's 2 am CET now (I just got back), I haven't eaten, and someone's definitely going to pay for this. I have shut down my old e-mail account, and I'm now using linuxdvd@mmadb.no - More information coming tomorrow, once I've talked to my lawyer. Did someone whisper countersuit?"

It was all a very sorry state of affairs, which did not help the MPAA's image one bit, and ultimately was doomed to fail. This was, of course, because by this point is was all too late, and the DeCSS code was copied all across the Net, and is still available in some surprising places. Now there are decryption programs based on DeCSS, such as *CaptainCSS*, which can be used in conjunction with some of the players reviewed here to view encrypted DVDs. All in all this whole saga was a complete waste of time.

This roundup of DVD playback applications for GNU/Linux will look at the main contenders – namely *Xine*, *VideoLan Client*, *MPlayer* and *Ogle*. In addition to playing encrypted DVD, unofficially in most cases, several of these players will also playback other file formats, which could be an added bonus.



The neat and tidy interface takes a bit too much desktop space for some people's liking.

Xine

■ **VERSION** 0.9.8 ■ **WEB** <http://xine.sourceforge.net/>

Xine has been around for a reasonable amount of time, gaining quite a large following. Like most of the players reviewed here it can play

unencrypted DVDs as well as additional file formats. Part of the reason for its popularity is because it supports plugins which allow you to

add support for addition codecs, both open and closed source, as well as other additions. These plugins are easily obtainable on the Net, and are simply to add to your Xine installation.

As mentioned, Xine only supports playback of unencrypted DVD out-of-the-box – to avoid any possible legal issues due to the legal problems of decrypting DVDs without the relevant

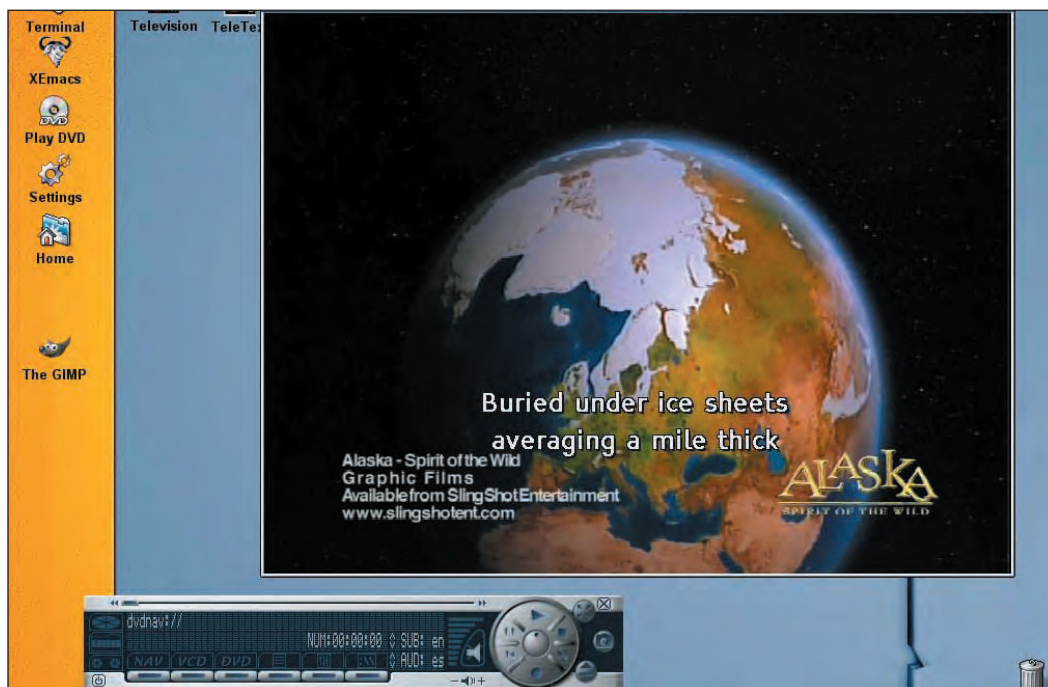
license. DVD decryption is available by using a third party plugin called *xine_d4d_plugin*. The legal status of this plugin is, as always, debatable, but it is easy enough to find and install.

There are some useful links on the Xine web site, and a search at Google should find the required links for this plugin. With the plugin, Xine's playback of encrypted DVDs is smooth – with no noticeable problems with audio synchronisation, or any video glitches.

Xine's interface is neat and tidy and provides most of the function you would require, but you may find that it takes up too much of your desktop space. As Xine is currently not skinnable there is no way to change the size or layout of the GUI. If you are running Xfree86 4.x with a supported video card, you can switch Xine from windowed to full screen mode and back again with the switching between modes being nice and smooth. If this is not supported by your video card, or version of X, Xine can still be viewed in a window, which is the default startup mode.

Xine does not directly support DVD menus, but there is a plugin called *Dvdnav* (available from <http://prdownloads.sourceforge.net/dvd/>) which adds this functionality to Xine. If you want to access all of the features of your DVD then this plugin is a "must install!" The plugin works very well, even with complex animated menus that some DVDs have, and although this is not required for DVD playback, it obviously gives you complete access to all the features available. The code for the DVD navigation was written referencing the original *Ogle* DVD menu code base, reviewed later.

Xine has grown into a justifiably popular, solid and very capable DVD (and other format) player. It is easily extended, making adding additional functionality far easier than having to recompile the source, which you have to do for most of the other players reviewed here. The only slight weakness is the interface which really needs to be improved and, preferably, support skins. But, all in all, worth looking into.



With support for menus and encryption, via its plugin architecture, Xine is justifiably popular.

LINUX Format VERDICT

Ease of use	10/10
Features	8/10
Performance	9/10
Stability	9/10

A very capable DVD player, with support for other formats and can access DVD menus via a plugin.

LINUX Format RATING

9/10

VideoLan Client

■ **VERSION** 0.2.92 ■ **WEB** <http://www.videolan.org/>



Top performance on lower specced hardware could sell you on VideoLan Client.

VideoLan Client (Vlc) forms a part of the VideoLAN project, designed to provide a full MPEG2 client/server solution. But you can also run *VideoLAN Client* as a standalone program to play MPEG2 streams from a hard disk or a DVD. To access encrypted DVDs *Vlc* uses the library

libdvdcss, which was developed by the *Vlc* project team referencing the original *DeCSS* code. *Libdvdcss* is a simple library designed for accessing DVDs like a block device without having to bother about the decryption. *Vlc* does not use *DeCSS*, but a different implementation which does not use

the cracked *Xing* decoder key.

DVD playback with *Vlc* is very smooth with no noticeable problems with audio synchronisation. When you start viewing a DVD it defaults to opening up a window to display the movie, but you switch smoothly to full screen mode by pressing the **f** key, in

the same way as the other players reviewed here. *Vlc* seems to provide the best performance on lower spec machines, which might be an important consideration.

The only slight problem with *Vlc* is the size of GUI, which is the largest of the group here. You can resize the GUI, but there is no way to start the GUI in a more minimalistic mode. Also, when you resize the GUI it does not scale down very well. The GUI itself is well laid out and easy to navigate with the usual button for play, stop, pause and so on. You can also access the preferences from the GUI, and change some setting to suit your installation, such as the path to your DVD device.

Vlc is a good DVD player and the best one here for low power machines. It does lack support for DVD menus so you cannot access any of the additional features of a DVD, so if you require this feature you will have to look elsewhere. However, DVD playback is as good as *Mplayer* or *Xine*, so if you can live with the overly large GUI, *Vlc* makes a capable DVD player.

LINUX Format VERDICT

Ease of use	8/10
Features	7/10
Performance	10/10
Stability	9/10

A good DVD player, ideal on lower spec machines but does miss out on DVD menu support and does not have very wide support for other formats.

LINUX Format RATING

8/10

Ogle

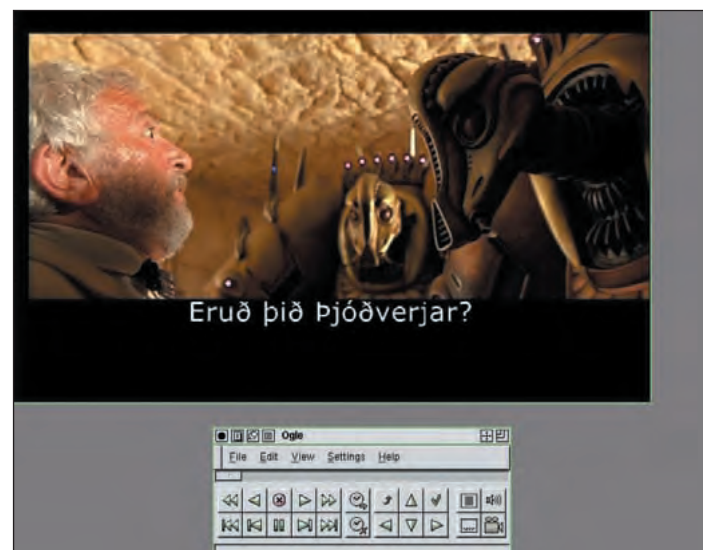
■ **VERSION** 0.8.2 ■ **WEB** <http://www.dtek.chalmers.se/groups/dvd/>

Unlike the other players here Ogle is a pure DVD player. It was the first to support DVD menus and navigation, the code of which is now used in the *Xine* plugin as mentioned earlier. As with *Vlc* and *Mplayer*, *Ogle* uses *libcss* and *libdvdread* to decode and read DVDs. The MPEG decoder features various levels of acceleration to take advantage of MMX processors and some hardware MPEG decoders.

Like *Mplayer*, *Ogle* can be run directly from the command prompt, but there is a GUI available if you prefer. Unlike *Mplayer*, *Ogle*'s GUI is not skinnable, but it is compact and

contains more functionality than *Vlc*'s GUI. However it is still larger than *Mplayer*'s default GUI and not as well laid out. There is a preferences option but currently this does not work.

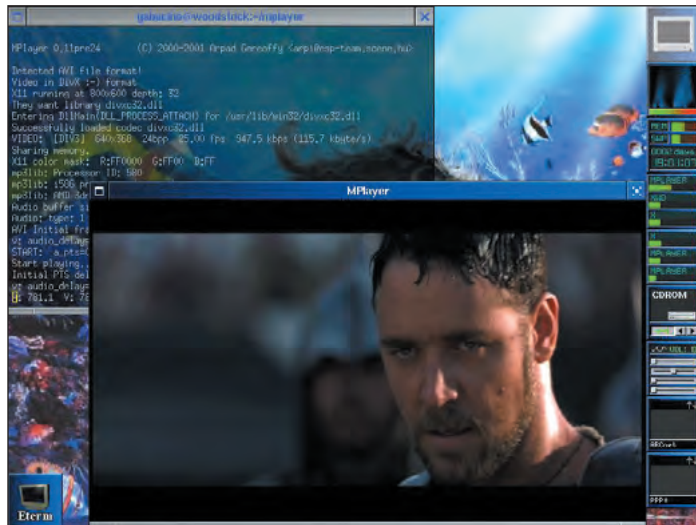
You can navigate the DVD menu by using the arrows on the GUI, but navigation using the mouse seems to be by far the easiest method. Unfortunately, playback of encrypted DVDs is not as smooth as with the other players here, with occasional freezes and audio glitches. However, this is occasional and does not detract too much from watching a DVD, but might be a consideration. As



The small but functional GUI is easy to navigate.

MPlayer

■ **WEB VERSION** 0.60 ■ **WEB** <http://www.Mplayerhq.hu/homepage/>



No matter what the format of the film, MPlayer is likely to support it.

MPlayer has by far the largest file format support of all the players reviewed here, in addition to encrypted and unencrypted DVD support. Currently Mplayer support most MPEG, VOB, AVI, VIVO, ASF/WMV, QT/MOV files, XAnim, and Win32 DLL codecs, and this list increases almost daily. In addition to this you can watch VideoCD, SVCD, DVD, 3ivx, and even DivX movies.

In addition, Mplayer also has support for a wide range of output drivers. It works with X11, Xv, DGA, OpenGL, SVGAlib, fbdev, AALib, and you can use SDL and some low level card-specific drivers (for Matrox, 3Dfx and

Radeon) as well. Most of them support software or hardware scaling, so you can enjoy movies in fullscreen. Lastly Mplayer supports displaying through some hardware MPEG decoder boards, such as the DVB and DXR3/Hollywood+. That's a very impressive list of support devices and file formats.

MPlayer also has one of the largest user communities – ensuring that development is likely to continue at the current rate.

With all this format support you might expect Mplayer to be slightly slower than the other players reviewed here, but this is not the case. Mplayer's

player performs very well, although there can be synchronisation problems with DVD playback on lower spec machines. However, if you have a fairly new PC (in excess of 500MHz), playback is as smooth as the rest of the players here. If you only have access to a machine with less power than this, Mplayer will not suit you as playback of DVDs can get very messy with audio synchronisation problems, jerky video and so on.

Encrypted DVD are supported using the libcss library and, optionally, libdvdread for chapter support. As with the other players, encrypted support is not directly provided by Mplayer. You will need to download the libraries yourself. Unlike Xine, Mplayer does not support plugins so you need to ensure that the libraries are installed before compiling. However additional codecs can be easily added by copying them into the relevant directory.

Mplayer, by default, is designed to be run from the command line – so if the command line is where you live, then this might be right up your street. However, if you do want to have the option of a GUI you need to add **--enable-gui** to the configure script before compiling. If you intend just to use Mplayer for DVD playback then there is probably no need for the GUI as Mplayer currently does not support DVD menus.

Even when you do compile Mplayer with GUI support, you need to download a skin for the GUI as Mplayer source does not contain one. Skins are easily downloaded from the Mplayer web site and from other sources. Also to run Mplayer with the GUI you need to either specify this

using the **-gui** switch, or link gmpmplayer to the Mplayer binary. The default skin is simple to use and also nice and small, so you do not lose a great deal of desk space.

Mplayer's main drawback – or at least irritation – is that you cannot access a DVD from the GUI, but instead you have to start Mplayer from the command line with the **-dvd** flag in order for it to play your DVD. Consequently, to view another DVD currently means that you have to restart Mplayer. Lastly, there is no support for DVD menus available at present, so you cannot access the addition features of your DVD with Mplayer.

As a one stop shop for movie playback Mplayer scores very highly. It is fast (assuming you have a fairly new PC) and DVD playback is very good with no audio synchronisation problems. With the support for multiple file format you may find that Mplayer is all that you need by way of a media player. However, if you want to have access to the DVD menu you will need to look elsewhere.

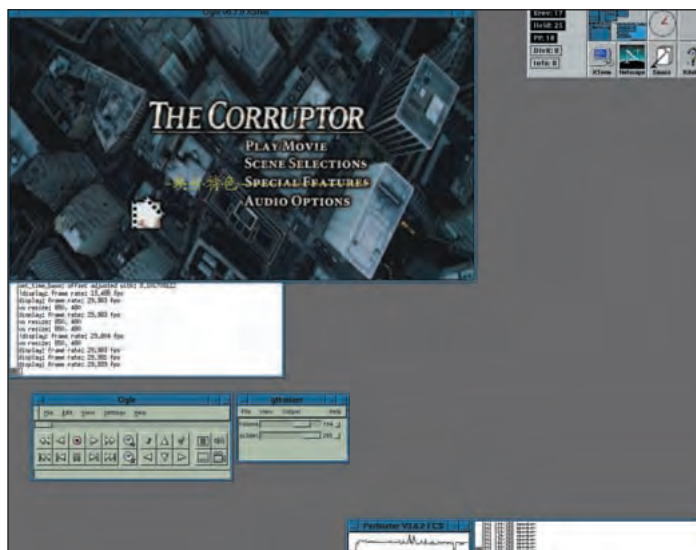
LINUX Format VERDICT

Ease of use	10/10
Features	10/10
Performance	8/10
Stability	9/10

Media player with knobs on, but needs a fairly powerful machine to run on.
The only down side is the lack of support for DVD menus.

LINUX Format RATING

9/10



Ogle – good at DVD playback, but that's all that it does.

with the other players reviewed you can switch between windowed and full screen mode, and back again, with the switching between modes being nice and smooth.

Ogle does have a few drawbacks, the main ones being that there is no chapter menu support, no angle selection during playback and no closed caption support. The most annoying issue is that you have to restart Ogle to play another DVD, which is the same issue Mplayer has. These may or may not be major issue to you, but, again, are worth taking into consideration.

Ogle is the only one of the players on review here that only plays DVDs, and no other formats. Its main claim to fame was the DVD menus support but, thanks to the fact that Ogle is OpenSource, the code base is now

being used in other players. If you only want to play DVDs then Ogle is worth reviewing, but if you needs are wider than that you probably want to look at one of the other players reviewed here.

LINUX Format VERDICT

Ease of use	9/10
Features	8/10
Performance	7/10
Stability	9/10

DVD-only player, that was special when it was the only one with DVD menu support, but Xine now has this capability.

LINUX Format RATING

8/10

DVD players **The verdict**

At last it is easy to view encrypted DVDs under GNU/Linux – using any of the players reviewed here, and there are sure to be more players that will become available over time. What could be better than to surf the web, or code, whilst watching *Hitch Hikers*? Yet, even though GNU/Linux is still not an officially supported Operating System for DVD playback, we probably have the best DVD players for any platform thanks to the Open Source model, and determined hackers such as Jon Johansen.

All of the DVD players reviewed here have come on leaps and bounds since they were last reviewed in *Linux Format*. *Xine* has probably improved the most, with many enhancements, such as DVD menu support, added to its impressive list of features. *Ogle* and *Mplayer* are interesting as they allow you to playback a movie from the command line without having to load up a desktop space consuming GUI, which you probably do not need if you only intend to view a movie. *Mplayer* has the biggest format support of all the players here, and the supported list increases almost daily. Although *Ogle* was originally going to be merged into the *Xine* project, this

‘Lacking only Xine’s DVD Menu support, MPlayer is the second most popular download from Freshmeat.net – beaten only by the Linux Kernel’

appears to have been put on hold for the time being.

It is interesting to see how much code is being reused across these players. As mentioned, *Ogle* was the first to support DVD menus and this code has been used to supply this function for *Xine*. The *Vlc* project produced the *libdvdcss* which is now

used by most of the other players here. It is this type of code reuse that make the Open Source programming model so powerful, and is good to see. This obviously leads to healthy competition between the various DVD player groups to make their player the best one available, and can only strengthen the quality of the source

code that they produce.

The performance of these players is fairly consistent across the board, assuming that you have a fairly modern computer – although *Mplayer* does perform poorly unless you have a PC running at 500MHz or more. *Vlc* seems to be the best performer for lower spec computers, but lacks several features, such as DVD menu support, that the other players provide.

Ultimately, the player that suits you best will depend upon your requirements, however the two that do stand out from the pack are *Xine* and *Mplayer*. Currently, *Mplayer* is quite a way ahead of *Xine* in terms of format support, the GUI is sensibly laid out (if you compile the GUI in), although it does not have DVD Menu support at present that *Xine* has via a plugin. If

they could intergrate the DVD Menu code from *Ogle* into *Mplayer* then this would be by far the best player for all purposes. However, at the moment if DVD menu support is important to you then the best option for a multimedia player is *Xine*. *Ogle* is also a fine DVD player if all you want is DVD playback, but its limited video format support means that you will need to install another multimedia player which is not necessary if you use *Xine* or *Mplayer*.

But do not just take our word for it. Using the Freshmeat popularity chart (<http://freshmeat.net/stats/#popularity>) as a guide, *Mplayer* is quite a way ahead with a score of 20,943, which makes it the second most popular project on Freshmeat, whilst *Xine* has a score of 11,605, which makes it the third most popular project there. Needless to say, the Linux Kernel is in first place, with a score of 30,171. But the size of the user community for *Mplayer* stands out.

So, if you are on the look out for a DVD players (with a few more features), first give *Mplayer* a go and if that does not suit, try *Xine*. One of these two players is bound to suit your own requirements. Happy movie watching. **LXF**



With a massive user community, and excellent format support, MPlayer edges out in front.

Table of features

Player	Full Screen	Encrypted DVD Support	Supports DVD Menus	DVD Accessible with GUI	Skinable	DVD playable from command line
Ogle	yes	External	Yes	No	No	Yes
Xine	yes	External	Via plugin	Yes	No	No
MPlayer	yes	External	No	No	Yes	Yes
Vlc	yes	External	No	Yes	No	No

Thin clients

If maintaining a large network of workstations is becoming difficult, try using thin clients instead, says **David Coulson**.

Offices, schools and universities, often need a large number of workstations available, usually with identical software and hardware setups. Employees, staff and students can log into any machine, and not notice any difference in the operation of the system. Maintaining software and, of course, ensuring that each machine works as expected, is quite a task, and even the most adept systems administrator will soon find themselves overwhelmed with work so that all of their time is spent updating machines.

cover feature



There are many different methods of keeping machines up to date and synchronising their configuration, but one of the most effective is the use of thin clients.

Architecture

As with anything of this nature, there are varying degrees of 'thinness' for your workstation, so you can adjust requirements depending on the budget, or lack thereof, which is available. Simply speaking, the less hardware you have client side, the more hardware and network bandwidth you need on the servers on the network. The general rule is to have many clients and a single serve – although, if you need to serve for a considerable number of clients, distributing load among a selection of servers would be preferred, as it's often cheaper to buy a number of mid-range servers, than a single high-end system, plus it adds in a little high-availability to the network.

The basic thin client is a system which does very little itself, all applications run on a remote server and all file storage is handled over a network using NFS. As one might expect, this is little short of incredibly network intensive, but it allows you to use very simple machines with low spec hardware as clients, although any servers must have considerable memory and CPU power at their disposal to handle everything which clients demand of them. With this style of configuration, it is very easy to add in new clients with little, if any, disruption to everyone else, and each machine behaves identically to any other as everything is running remotely, and is unaware of what it is really running on.

At the other end of the scale, we have fairly powerful workstations, running all applications locally, but having application binaries and user home directories mounted over NFS. This drastically reduces the amount of network bandwidth used, as for basic system binaries the system can simply access a local hard disk, but for running any applications, or for accessing files from within a user's home directory, it will make use of the networking capabilities. The *caveat* here is that you need better hardware for the clients, but if your users are running very CPU or memory intensive applications, then it is quite easy to justify this change in

architecture. This requires a little more maintenance, but if you use a distribution such as Debian, which facilitates network based package management, then it is a breeze to keep the workstations up to date.

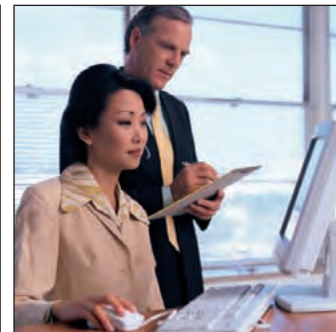
It is, naturally, perfectly possible to have a network which has a mixture of both of these systems, yet everything for the users remains consistent throughout. You may need more powerful workstations running local applications for graphics and application development, but for people who only need to run web browsers or office applications, a remote execution system is a desirable option. You may also choose to have a mixture of both remote execution and local applications on some systems, so you don't have wasted resources.

A family history

Thin clients began as simple text based terminals, connected via serial interfaces to a large mainframe. Back in the days when few people were allowed to see a real computer, let alone touch one, a terminal was the only way to get things done on a computer. Much like a console on a Linux system, you could run simple character-based programs on it and, indeed, many well known applications, such as *Emacs*, were developed purely for use on terminals. Fortunately, computing has progressed since then, and we're in a world of GUIs, mice and multimedia, which makes life a little more complicated. Many companies

are producing thin client equipment, such as the famous ARM-based Netwinder system, which is little more than a keyboard, mouse and display, connected to a very small embedded system which has a network interface to the outside world. There is no reason why you could not use older hardware, such as original Pentium or K6 systems, as long as they had sufficient network capabilities to function as required.

You'll find running X applications over a 9600baud serial interface nothing short of annoying and stressful, so a proper Ethernet based network is needed to get things done properly. For simple X applications with a few clients, 10BaseT or 10Base2, will be adequate, but once you move into more complex interactive applications, such as graphics or design packages, or web browsers with Flash or other animations, you'll soon find your network crawling, and any other



Thin client architecture simplifies administration of end user's boxes.

“There are varying degrees of ‘thinness’ for your workstation, so you can adjust requirements depending on the budget.”

operation which needs to be performed on the same network will suddenly drag like a snail. With the cost of network equipment being so low now, it's difficult not to justify



Swap shop

Swapping to disk is much cheaper than buying more RAM, but what if you have no disk?

If you're going to run local applications on a client, then you need as much RAM available as you can muster. But without a local hard disk, swapping pages of memory out of RAM to the swap file is rather difficult. Swapping over NFS is, as default, denied, as it is considered a rather pointless use of network resources – and, if you're swapping a lot, then it can slow the network to little more than a pathetic crawl. However, since we have little option, we need to allow NFS to permit us to enable a swap file on a remote machine. As usual, there is a kernel patch for the client, which is easily applied and will let you **swapon /mnt/nfs/swap/swapfile** without causing any problems. One *caveat* is that you must ensure that no two systems are

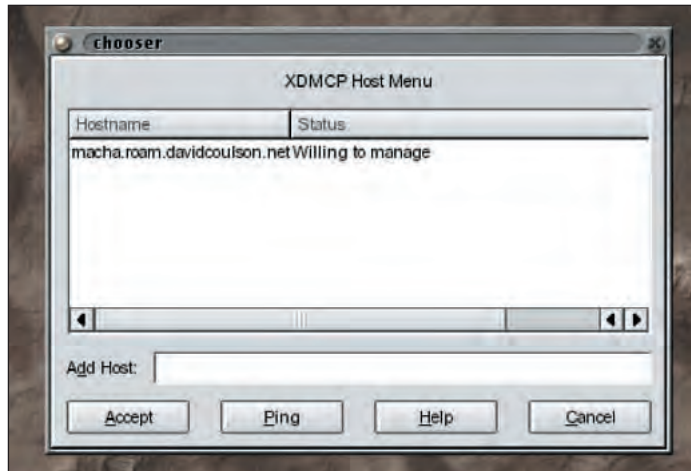
using the same swap space at the same time, as this will cause all sorts of interesting problems, so storing swap files as 'swap/<ip>', assuming all IP allocation is controlled via DHCP, and then having the machines **swapon** the appropriate image, will make the whole process much less painful.

This does have a number of pitfalls, as you would expect. One major consideration, especially on a public network, is that NFS is unencrypted and people could, in theory, sniff the swapped pages – which could contain all sorts of information which is being held by processes on the system. There is little you can do about this, other than buy a physical memory for the machines or drop in an old hard disk, to be used purely for swap space. You also need to

protect against memory fragmentation by limiting the size of the packets which NFS uses. IA32 systems have 4KB pages, so if you are swapping and you transfer 6KB at a time, you are not always transferring an entire page and it will store the extra transferred data in memory, which isn't clever when you've not got any left. Reducing this to 2KB will ensure that you will never overrun with extra swapfile data, although this disables asynchronous I/O, so you want a single mount for swap files and nothing else, otherwise performance for regular files will be dismal at best.

The nfs-swap kernel patches, along with documentation can be found at <http://www.instmath.rwth-aachen.de/~heine/nfs-swap/nfs-swap.html>

ThinClients



The X chooser applications offer you a list of remote XDMCP servers which you can log into.



installing 100BaseT hardware, or even going so far as to have 1000BaseT uplinks to the various servers on the network. It's quite unlikely that a single machine could saturate 100Mbit by itself, but if you have twenty clients, all using 10Mbit, you've suddenly gone 100% over the limit.

Configuration

A thin client is configured to download a kernel from a server, then proceed to boot as a normal machine. Booting the client is the simplest stage, as you only need to ensure that your client has a NIC with an Etherboot ROM on it – so that it will boot from the network, rather than using a local



LTSP uses XDMCP to give remote login capabilities to the machines connected to the server via the LAN.

device. Booting from a network isn't quite as simple as booting from a hard disk or CDROM, as you would expect, and there are a number of stages involved within the process in order for you to end up with a machine which can then boot a kernel. First of all, the Etherboot ROM will initialise the card, so it negotiates a link with the hub or switch, then proceeds to perform a DHCP request to the local network. DHCP is a protocol for dynamic configuration of a network, using information stored on the server, such as the range of IPs which can be used, and any DNS and routing information required for the client to access the network properly. As part of this, you can also include extra information, which we will need, including the location of a kernel image and any parameters for it. DHCP does not handle the kernel image itself, and instead, this is passed to a *Trivial File Transfer Protocol (TFTP)* server, which is

commonly used on networks to upload configuration and ROM images to network equipment. Once the Etherboot ROM on the client has downloaded the kernel via *TFTP*, it will then boot it, and everything from now on is controlled by the kernel image and it will proceed to start up like any other Linux system. It does need to perform a number of basic operations, such as performing another DHCP request, so that the *dhcpclient* system within Linux is aware of the lease which the client has. Mounting the root file system is the fun part, as you will generally do this over NFS and then proceed to launch any required servers or client applications as appropriate to the default run level of the system. One of the most basic problems of this setup is that you need somewhere to write temporary files, which is specific to the client. Generally, the root file system is mounted read-only, and then */home* is mounted over that with read/write

“In the days when few people were allowed to see a real computer a terminal was the only way to get things done.”

Block device alternatives

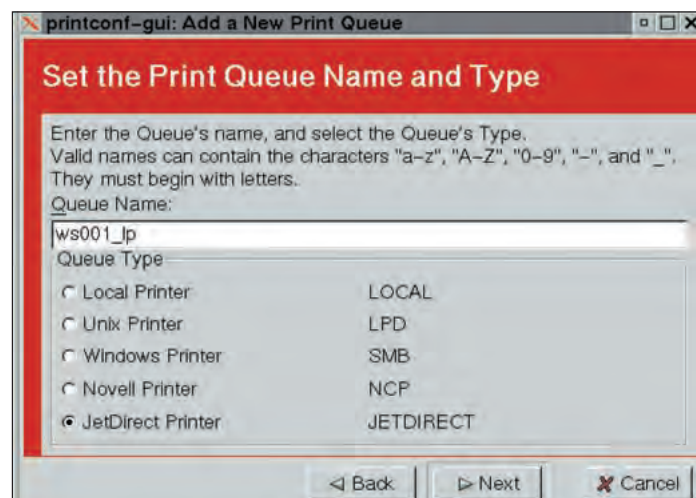
NFS for swap space is rather slow, but NBD is an alternative

Swapping over NFS is an option, but having a system designed purely for high-I/O file transfer over a network is much cleaner and more efficient. However, it has numerous *caveats*, so another option is to make use of the **Network Block Device** kernel module. This, by use of a client and server daemon, will permit you to access */dev/nb0* as if it were a local block device – so you could mount it like a hard drive, or, more crucially, convert it to a swap file and use it purely for swap space.

The user-space utilities for network

block devices function, and transfer data, over TCP, so employing socket encryption is fairly straight-forward using either *ssh* or *stunnel*. This makes it somewhat more desirable for applications which will be handling sensitive data, although it may be desired that networks be segmented depending upon the type of usage of the various machines.

NBD is included in all 2.2 and 2.4 kernels, but you will need user-space utilities which can be downloaded from <http://atrey.karlin.mff.cuni.cz/~pavel/nbd/nbd.html>



Printing with LTSP is a breeze, and clients can be used as print stations.

permissions, but we also need /tmp and /var, which requires a little creativity. A small RAM disk, 16MB or so in size, is often used here, and is mounted under /tmp, and any services or scripts which write to /var are rewritten to write to a /tmp/var/ directory instead. This, of course, takes 16MB of memory away from your client, but if the majority of your applications are going to be running remotely, then it will not be a problem.

Once the machine comes up happily, it's entirely up to the configuration of the various services running on it, such as *xdm*, as to how things are handled from here. We can run applications in local memory by running them from within a NFS mount, or we can run them remotely using *rsh* or *ssh*, so we have quite a number of choices depending upon hardware specifications available to us.

The joy of X

As we know, X is a very useful protocol, as it is completely network portable. Clients generally don't care if the display is on the local machine or at the other end of a network interface, which makes X a perfect platform for a thin client GUI. Not surprisingly, X isn't exactly the most bandwidth efficient protocol, by its very nature, so if you want everything to run smoothly and you don't want problems with people waiting for their applications to redraw every time they move a window around, then you're going to need 100BaseT, at least, if not something more should you want to use real-time applications.

X clients function by running a XDMCP server, such as *xdm*, on our server, then executing the X server

Etherboot

Booting from a NIC is easily done, but what if you don't have a boot ROM?

Most NICs have a socket for a boot ROM, but unless you specifically buy a NIC which has the ROM attached, you'll be lucky if it does. Boot ROMs are EEPROMs, generally 16, 32 or 64KB in size, and store a boot image which is used to start up the network card and perform DHCP requests. Many manufacturers have boot images for their NICs available, but often they

don't, so what can you do?

Fortunately, a nice group of people are working on the 'Etherboot' project which focuses on the development of boot ROMs for popular NICs. You, of course, need an EEPROM writer to upload the image to the ROM.

EEPROM writers are available from the majority of electronics stores, other than that, anyone can create

their own boot images for NICs.

Etherboot is available at <http://etherboot.sourceforge.net/>, which has extensive documentation covering the use of BOOTP/DHCP in order to have a disk-less workstation, as well as many suitable

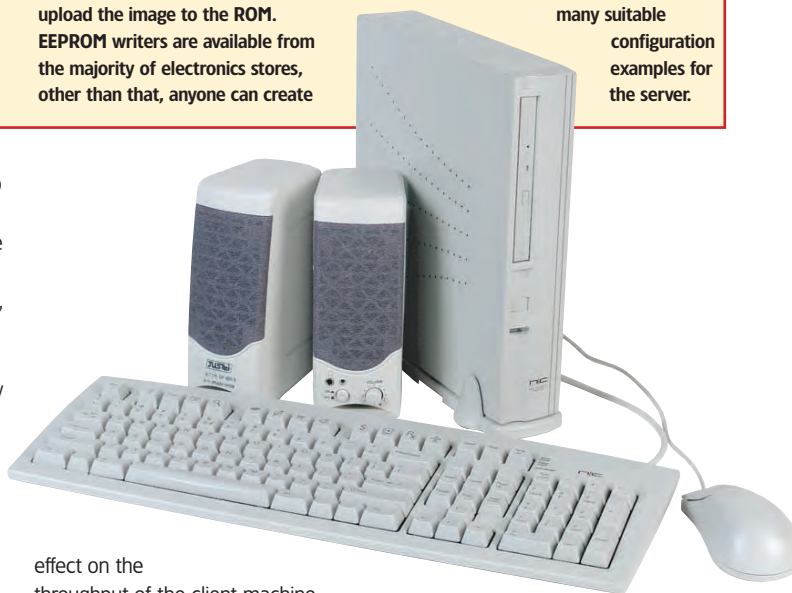
configuration examples for the server.

locally on the client and connecting to the XDMCP service for login and window manager execution. When the X server starts up, it will broadcast a XDMCP request to the entire network, and the *xdm* service will respond and setup the X session appropriate to its config. While we have a local X display running, the *xdm* login session, and anything spawned from that – such as the window manager and apps, not to mention *xterms* – will be running on the remote server and all of the X display information will be passed over the network to our local machine. Naturally, this is great if your local system couldn't run *FVWM*, much less *Star Office* or *KDE*, but it does put quite a load on the server, particularly if you have many clients accessing it at one time, not to mention that your network will be getting quite a thrashing every time someone even moves a window.

An alternative is to keep the XDMCP system, yet run some applications locally on the client. Even running *xterms* or simple applications locally will reduce network bandwidth considerably, without having a bad

effect on the throughput of the client machine. We have to employ either *rsh*, and then change the **\$DISPLAY** variable in our terminal, or make use of the X port forwarding capabilities of *ssh*. Ironically, as our X session is remote, we need to *ssh* from the remote system back into our workstation, then run the applications. This can be easily automated, with the use of *ssh keys*, or *ssh-agent*, so that the user can transparently run applications from any machine on the network.

Since our X terminal server will be seeing quite a thrashing, it's often best to separate that from the file server, so that high-bandwidth X applications do not slow up general file transfers by anything else. Of course, the X server will need access to the same files as any other workstation, such as /home, so one can either setup a separate physical network between the X server and the file server, or employ the use of traffic shaping to ensure there is enough bandwidth available for NFS, should anyone use up all of the available bandwidth to the X server. It takes a little time and understanding to decide how best to architect your network to make the best of your available bandwidth, but thin client servers are easily set up and separated. If you need to you can very easily add or remove hardware at



Larry Ellison's New Internet Computer – not the world's thinnest client, but flexible.

```
david@tailtiu:~ (pts/0)
default-lease-time 36000;
max-lease-time 720000;

subnet 10.1.3.0 netmask 255.255.255.128 {
    range 10.1.3.64 10.1.3.127;
    option broadcast-address 10.1.3.128;
    option routers 10.1.3.1;
    option domain-name-servers 10.1.1.5;
    option domain-name "dnz.davidcoulson.net i.davidcoulson.net davidcoulson.net";
}

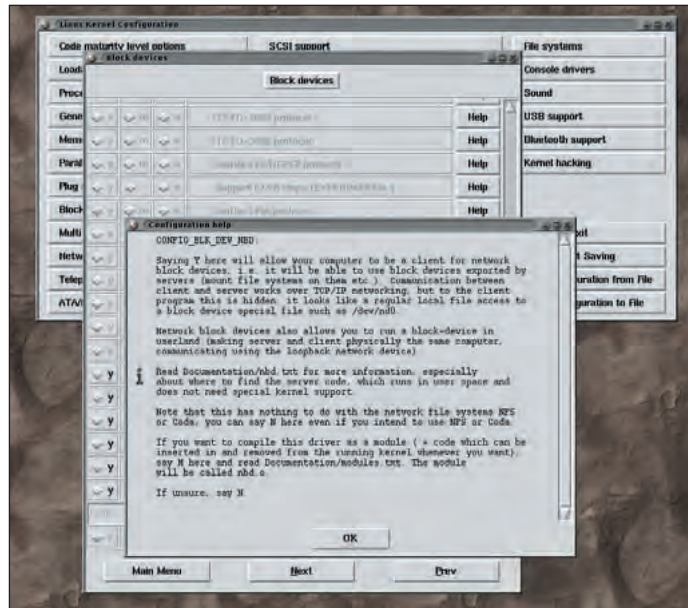
host macha {
    hardware ethernet 00:00:46:10:a6:04;
    fixed-address 10.1.3.7;
}

subnet 10.1.3.252 netmask 255.255.255.252 {
    option routers 10.1.3.253;
    option broadcast-address 10.1.3.255;
    option domain-name-servers 10.1.1.5;
}
```

DHCP configuration can be simple, or complex, but with thin clients you need extra options.



ThinClients



The Linux Network Block Device will help in enabling network swapfile capabilities for your clients.

will without disrupting the whole network very much.

Easy way to do it

Setting up all of these services, and configuring them all the same, can be quite a task, so if you're after a very quick and clean method to setting up a thin client server, then look no

further than the *Linux Terminal Server Project*, or *LTSP* (www.ltsp.org). Simply, *LTSP* configures all of the services, such as *dhcpcd*, *tftpd*, *xdm* and so forth on your server, with a single script, and has extensive client kernel capabilities and useful scripts for ensuring your clients come up happily.

LTSP has four packages; The *LTSP*

core, the *LTSP kernel* for the clients, the *X core* and *X fonts* for clients. If you don't need X on the clients, then you can skip the last two packages, and if you intend for your X server to serve fonts via *xfs*, then you can skip the fonts package too.

LTSP initialisation is simple:

```
# cd /opt/ltsp/templates ;
./ltsp_initialize
```

It will ask you what you want available on your server, so if you're building two *LTSP* servers to sit on the same network, you'd want to exclude certain services from running twice, but you have the options of;

- *XDM* - X Display Manager
- *GDM* - GNOME Display Manager
- Display manager startup script
- *bootp*
- NFS /etc/exports file
- *tcpwrappers*
- Port mapper
- *syslogd*
- *TFTP* startup script

Note that *LTSP* does not touch NIS or named based configurations, so if you want to distribute user/password information across a network, or have a proper DNS zone for your machines, then you will need to do that by hand.

DHCP is the first service to configure, as this is the basic system used to actually start your thin client off properly. You need to pick an IP range for your network, and *LTSP* defaults to 192.168.0.0/24, so your machines will use 192.168.0.1 through to 192.168.0.253, as 192.168.0.254 is reserved for the DHCP server; If you've got another *LTSP* system, then you might want to call that 192.168.0.253.

We can allocate IPs either dynamically, on a first come, first served basis, or we can assign IPs based upon the MAC address of the client. The latter is usually a better option, then you can assign host names to machines, plus you always know which IP a specific box will have. An example /etc/dhcpd.conf configuration for *LTSP* may look something similar to the following;

```
default-lease-time 21600;
max-lease-time 21600;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers 192.168.0.254;
option domain-name-servers 192.168.0.254;
option domain-name "ltsp.org";
option root-path
```

LTSP Configuration Options

Power and control from a text file

LTSP has a great number of configuration options, in order to enable various additional features on the system. Many of them will not be used, but a lot are very useful;

LTSP_BASEDIR - This indicates where the *LTSP* root file systems are located. The default value is /opt/ltsp

SERVER - This is the server that is used for the **XDM_SERVER**, **TELNET_HOST**, **XFS_SERVER** and **SYSLOG_HOST**, if any of those are not specified explicitly. If you have one machine that is acting as the server for everything, then you can just specify the address here and omit the other server parameters. If this value is not set, 192.168.0.254 will be used.

SYSLOG_HOST - If you want to send logging messages to a machine other than the default server, then you can specify the machine here. If this parameter is **NOT** specified, then it will use the **SERVER** parameter described above.

NFS_SERVER - This specifies the IP address of the NFS server used when the /home file system is mounted. The

default is whatever the **SERVER** is set to.

USE_NFS_SWAP - Set this to **Y** if you want to turn on NFS swap. The default is **N**

SWAPFILE_SIZE - This is how you can control the size of the swapfile. The default is 64MB.

SWAP_SERVER - The swapfile can exist on any server on the network that is capable of handling it. You can specify the IP address of that server. The default is whatever the value of **NFS_SERVER** set to.

NFS_SWAPDIR - The directory on the server that is being exported via NFS. The default is /var/opt/ltsp/swapfiles. Make sure that the directory is being exported in the /etc/exports file.

TELNET_HOST - If the workstation is setup to have a character based interface, then the value of this parameter will be used as the host to telnet into. If this value is **NOT** set, then it will use the value of **SERVER** above.

DNS_SERVER - Used to build the resolv.conf file.

SEARCH_DOMAIN - Used to build the resolv.conf file.

RAMDISK_SIZE - When the workstation boots, it creates a ramdisk and mounts it on the /tmp directory. You can control the size of the filesystem with this parameter. Specify it in units of Kbytes (1024 bytes). To create a ramdisk of 1 MB, specify **RAMDISK_SIZE = 1024**

If you change the size of the ramdisk here, you will also need to change the size of the ramdisk within the kernel. This can be compiled in, or if you are using Etherboot or Netboot, you tell the kernel the ramdisk size when you tag the kernel with **mknbi-linux**.

LOCAL_APPS - If you want the ability to run applications locally on a workstation, set this variable to **Y**.

NIS_DOMAIN - If you do setup **LOCAL_APPS**, then you need to have an NIS server on the network. The

NIS_DOMAIN entry is where you specify the NIS domain name. It needs to match a domain name that has been defined on the NIS server. Default value is **ltsp**.

NIS_SERVER - Set this to the IP address of your NIS server if you don't want it to send a broadcast looking for an NIS server.

Printing using LTSP

LTSP lets you take advantage of any printing device connected to your client.

As well as being a GUI terminal, LTSP allows you to use the thin client hardware as a print server. Rather than connecting a selection of printers to your server, which is probably hidden away in a back room somewhere, you can hook a printer up to your little workstation and allow all of your LTSP clients to print through it.

You need to edit the `/etc/lts.conf` file, and add **PRINTER** entries for the appropriate workstation to which the

printer is connected;

```
[ws001]
PRINTER_O_DEVICE = /dev/lp0
PRINTER_O_TYPE = P
```

This tells the `lp_server` program which port the printer is listening on, and what type of connection it has, serial or parallel. You can then use **printtool** to add a print queue on the server, accessing the printer over TCP/IP, and emulating a HP JetDirect printer.

```
"192.168.0.254:/opt/lts/i386";
shared-network WORKSTATIONS {
    subnet 192.168.0.0 netmask
    255.255.255.0 {
    }
}
group {
    use-host-decl-names on;
    option log-servers 192.168.0.254;
    host ws001 {
        hardware ethernet
        00:E0:18:E0:04:82;
        fixed-address 192.168.0.1;
        filename "/lts/vmlinuz.lts";
    }
}
```

We've given the machines six hourly leases – which is enough to ensure that leases are not held for machines which are not there, but we're not wasting a whole load of network bandwidth on pointless DHCP requests. 192.168.0.254 is our DHCP server as well as our NFS server and default route to the outside world. Our clients will mount the `/opt/lts/i386` directory on the server as their root file system, and use `/lts/vmlinuz.lts`, from our TFTP server for their kernel image. Note that LTSP has the tftp server running in secure mode – so all paths are relative to the `/tftpboot` directory, rather than the root file system of the server.

LTSP comes with two kernels as default; one which contains the *Linux Progress Patch (LPP)*, so that you have a much prettier start up of the client, and another which will display the usual boring text based boot information. Generally the latter is more desirable until you're 100% sure there are no config problems, as it's much easier to watch the text output from the kernel, rather than rely on the LPP system to inform you appropriately, if at all.

Within our `/etc/dhcpd.conf`, we've referred to the machine with the IP '192.168.0.1' as 'ws001', but we also need to provide host name resolution, so we can access the client by the name 'ws001' as well as the IP. This requires an addition to `/etc/hosts`;

```
192.168.0.1 ws001
```

As you would expect, you need to do this for every machine on the network, so it can take a while. If you prefer to use *BIND* to handle DNS, then you can use a **\$GENERATE** line in your zone to automatically generate all the A records, rather than typing them all out. Although you do need to remember to include reverse DNS entries for the **0.168.192.in-addr.arpa** zone.

Last but certainly not least is the `/etc/lts.conf` file for our clients, which we edit on our server as it is stored there within the `/opt/lts/i386` root directory. This controls the runlevel of our clients, the X config as well as various miscellaneous parameters for swapping and application access. An example `/opt/lts/i386/etc/lts.conf`:

```
# Config file for the Linux Terminal
# Server Project (www.lts.org)

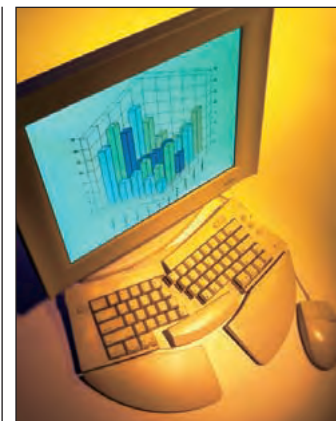
[Default]
SERVER = 192.168.0.254
XSERVER = auto
X_MOUSE_PROTOCOL = "PS/2"
X_MOUSE_DEVICE = "/dev/psaux"
X_MOUSE_RESOLUTION = 400
X_MOUSE_BUTTONS = 3
USE_XFS = N
LOCAL_APPS = N
RUNLEVEL = 5

[ws001]
X_MOUSE_PROTOCOL = "Microsoft"
X_MOUSE_DEVICE = "/dev/ttyS1"
X_MOUSE_RESOLUTION = 50
```

```
X_MOUSE_BUTTONS = 3
X_MOUSE_BAUD = 1200
USE_NFS_SWAP = Y
SWAPFILE_SIZE = 48m
RUNLEVEL = 5

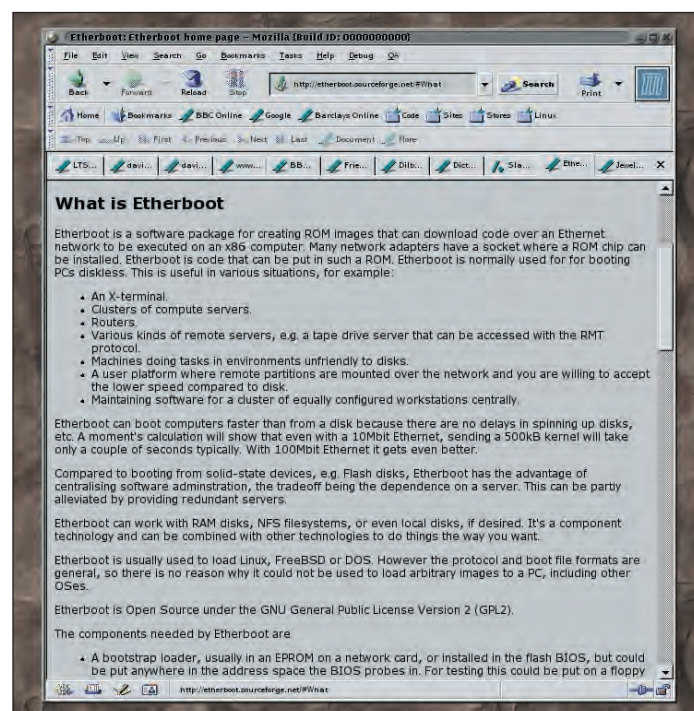
[ws002]
XSERVER = XF86_SVGA
LOCAL_APPS = N
USE_NFS_SWAP = Y
SWAPFILE_SIZE = 64m
RUNLEVEL = 3
```

Setting up a thin client can be an interesting and educational process and, of course, it gives you another reason not to throw out the old kit when you can do something useful with it. There are so many permutations of configurations based on hardware and network capacity that one can literally create a unique and powerful thin client layer on one's LAN, and simply plug in machines to take advantage of it. With the advent of Microsoft's .NET service, and the Open Source variant for GNOME, there is little wonder why thin clients are becoming a very popular network model for businesses. [LXF](http://www.linuxformat.co.uk)



It can look pretty, but needs little in the way of processing power.

“Rather than having printers stuck in the server room you can hook one up to your little workstation for all the LTSP clients.”



The Etherboot project (etherboot.sourceforge.net) will construct ROM images for most common NICs.

What on Earth are... NEW NETWORK FILESYSTEMS?

Charlie Stross introduces a better way of networking filesystems.

» So what is a network filesystem?

A filesystem is a way of arranging and locating files. Normally, we work with filesystems that help us find data on our own machine's disks. A network filesystem is somewhat different: it looks like a normal filesystem to the user, but rather than locating data on a local disk, a network filesystem talks to another computer, via a network, where the files are really stored. Usually there's a perfectly ordinary disk filesystem on the server's hard drives; this is *exported* via a network filesystem. Client machines then *import* the remote filesystems, and mount them, so that their users can work with files on the remote machine as if they're stored locally.

The big difference between a network filesystem, and a network-based file transfer system like FTP or the world wide web, is that programs that works with files over a network filesystem uses the same functions to gain access that they'd use to edit a local file: to work with a file via FTP or WWW requires a different set of tools and provides less functionality. (You can load or save a file via FTP or WWW, but you can't easily put a lock, or update just the 10,000th byte, in a file, for example.)

» What are the standard network filesystems today?

The first really widespread network filesystem was Sun's NFS, which is (clearly) an acronym for Network File System. NFS was introduced by SunOS and ported to other versions of BSD UNIX in the mid-1980's. NFS runs over TCP/IP and UDP/IP, the internet protocols. At roughly the same time, Novell

Netware began gaining ground as a network filesystem in the MS-DOS world, and Apple introduced *AppleShare*, a file sharing layer running on top of the *AppleTalk* networking system. Microsoft acquired the NetBEUI system late in the 1980's, then in 1991 introduced SMB, a file and print sharing mechanism on top of NetBEUI networking.

Originally, many of these network filesystems ran on top of networking protocols other than IP. However, since 1995 – when the Internet exploded into public view – virtually all of them use IP-based protocols for transport. For example, *AppleTalk* now runs almost exclusively over IP, as does SMB; NFS did so from the beginning.

» How useable are the old network filesystems with Linux?

Linux talks to just about *everything*. It can export filesystems to Macs, Windows clients, and other UNIX-type systems; it can mount filesystems from UNIX or Linux systems, Windows, Netware, and (with a little work) MacOS. This contrasts radically with the picture on other operating systems, which generally only talk their own proprietary protocols.

Linux is a server OS. NFS, as the native UNIX network filesystem, is well-supported – although threading issues in the kernel prevented versions of Linux prior to 2.4 from giving good performance. Today, Linux is a robust NFS server platform, with NFS server support built into the kernel; Linux systems can also mount remote filesystems easily via NFS. You can find out how this works in the NFS-HOWTO document (available from

<http://www.opencontent.org/openpub/>).

For dealing with Windows clients, the *Samba* package provides a comprehensive SMB file service (see <http://samba.org/> for a master list of *Samba* web sites and links to documentation and software). There are several books about *Samba*, and enough online documentation to fill this magazine several times over. SMB isn't a native UNIX-based filesystem, but by using the *smbmount* program you can mount SMB filesystems served by other hosts.

For dealing with Novell networks, the NCP remote filesystem protocol is supported; you can use *ncpmount* to mount volumes from a Netware fileserver, and also use Netware print queues.

For dealing with Macs running versions of MacOS prior to MacOS X (which, being BSD UNIX based, understands NFS), Linux has a package similar to *Samba*: *Netatalk*. *Netatalk* provides support for the *AppleTalk* network protocols as a layer running on top of the IP protocol stack. It also provides file and print services for Macs by way of the *atalkd* *AppleTalk* daemon. If you work in a Macintosh environment and need to mount *AppleShare* volumes on a Linux client system, you need to install the *AFPPS* package, (from <http://www.panix.com/~dfoster/afpps/>).

» What's wrong with the traditional network filesystems?

Lots. All of these filesystems share a number of problems. Firstly, they all rely on a single server. If you're working on a client machine and you've mounted a filesystem, and you're editing a file, then if the server crashes for some reason you're in big trouble – you won't be able to save your work (or you may

completely lose the file). While this isn't a huge problem for small networks, if your company wants to consolidate all its data onto a storage area network, this is bad news – a single machine failure can stop the entire company dead in its tracks.

We could ignore this problem if the filesystems provided one extra feature, called disconnected operation – basically, if the server is inaccessible, any writes by the client are stored locally: then, when the server becomes available again, the filesystem will handle the job of updating everything and handling any problems (e.g., two people working on the same file and saving updates while disconnected). It also usually entails having the client cache recently accessed remote files and directories, so that the client can continue to 'see' an image of the remote filesystem, even when disconnected. This is in contrast to NFS, *AppleShare*, or SMB – if you lose a connection to a shared drive or exported folder, it just goes away, leaving a hole in your filesystem.

At the highest end, there are network filesystems with multiple servers. Each server retains its own copy of the exported filesystem; by passing messages whenever a client updates a file on one of the servers, they try to stay in sync. This provides real disaster proofing – a server can go up in smoke in your machine room, but the company's data will not only be safely replicated – users working on it at the time won't notice anything happening. (Unless they're evacuated by the fire brigade!)

Clustered servers are overkill for small businesses or home users, but disconnected operation isn't. If you have a laptop and a desktop PC, you probably want your mobile machine to see the same home directory as your desktop, even when they're not connected. It's a real pain to manually copy files from one machine to another: by designating the desktop machine as a

server and using a network filesystem that supports disconnected operation to import your home directory, you can make this work. Disconnected operation is also useful for clusters of machines that are serving web files for a business, and need to provide the same data even when one machine is down for maintenance.

»» What are AFS and Coda?

There are a number of experimental multi-server network filesystems supporting disconnected operation on the client side. The best known is *Coda*, from CMU (www.coda.cs.cmu.edu/). *Coda* is designed to support server replication, so that a cluster of servers can stay in lockstep, and continue operation, even when the network is suffering a partial failure. Each *Coda* client runs a cache app called *Venus*; this communicates with the kernel, and answers queries for data either from a local cache file or by requesting a copy of the file over the network.

The other major contender is *OpenAFS*, from IBM. *AFS* was developed as a commercial product by IBM; *OpenAFS* is a version of *AFS 2* that IBM has donated back to the Open Source community under the IBM Public License. The source code release is available from <http://oss.software.ibm.com/developerworks/opensource/afs/>. It's not yet compiling under Linux, and is not yet patched into the kernel, but it promises an industrial-strength distributed filesystem with backup tools, ACLs, replication support, and secure transactions; expect it to show up in the 2.4 kernel within the next six months.

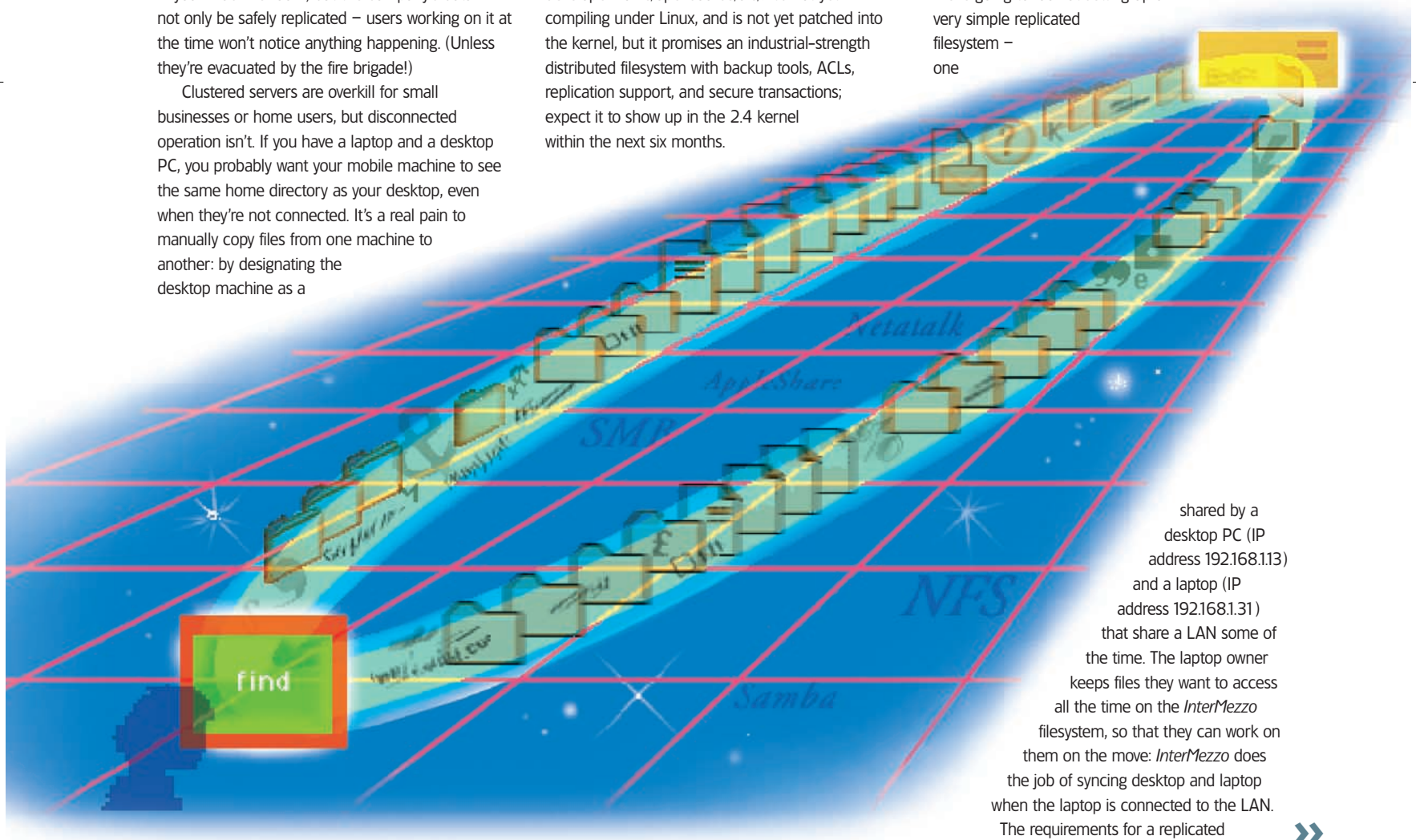
»» What about InterMezzo?

InterMezzo (from <http://inter-mezzo.org/>) is a clean-sheet design inspired by *Coda*, but stripped down and simpler. Its goal is to provide support for flexible replication of directories, with disconnected operation and a persistent cache. *InterMezzo* uses an existing *ext2* filesystem as the storage location for all data. When an *ext2* file system is mounted as type *InterMezzo*, instead of *ext2*, the *InterMezzo* software starts monitoring all access to the file system. It manages the journals of modification records, and negotiates permits to modify the disk file system, to avoid conflicting updates during connected operation. When disconnected, the contents of the *ext2* filesystem are still accessible to the client. There are two components; the *Presto* kernel module (which handles journaling and permit negotiation) and the *Lento* server (which replicates file system information between hosts).

InterMezzo is currently under development and doesn't yet have a resynchronization mechanism (to deal with situations where the client and server have inconsistent caches), but this is under development.

»» What do I need to know before I try installing InterMezzo?

We're going to look at setting up a very simple replicated filesystem – one



WhatOnEarthNetworkFS

filesystem using *InterMezzo* are that both systems should support the *ext3* filesystem (used for storing files on physical disks) and *InterMezzo* (used for network synchronization). *InterMezzo* in turn relies on *librsync*, a package for network synchronisation (which may not be installed by default on all distros). Do not confuse *librsync* with the (much commoner) *rsync* utility! This is *not* normally bundled with most Linux distributions, and the requirement isn't made very clear in the *InterMezzo* HOWTO. You will need to download and compile *librsync* (but not *rproxy*) from <http://rproxy.sourceforge.net/>.

InterMezzo isn't secure, so if you're going to use it anywhere except on a private LAN you should also have *ssh*, the *secure shell* – *InterMezzo* is designed to use *ssh* tunnelling where available. *InterMezzo* has been merged into the Linux kernel source tree as of kernel 2.4.15. If you have an earlier kernel, you will need to download the *InterMezzo* module source code and compile it, following the directions in the HOWTO. on the *InterMezzo* website. If you are running Red Hat 7.1 or 7.2 you are in luck: there are precompiled RPMs of the *Intermezzo* module and software at <ftp://ftp.inter-mezzo.org/pub/intermezzo>.

How do I install the InterMezzo software?

Compiling *InterMezzo* from sources is a pain: here's how we did it on a system running SuSE 7.3.

Become root. to begin building the programs *InterMezzo* relies on, starting with *librsync*:

```
cd /tmp
wget http://prdownloads.sourceforge.net/
rproxy/librsync-0.9.5.tar.gz
```

```
tar xvfz librsync-0.9.5.tar.gz
cd librsync-0.9.5
./configure --prefix=/usr && make install
cd /tmp
```

Now to install the essential Perl modules, including **POE**, an event-driven dispatcher used by the *Lento* cache manager, and **XML::Parser** & **XML::Simple**, used to parse *InterMezzo*'s XML configuration files:

```
perl -MCPAN -e 'install POE;
install XML::Parser; install XML::Simple;'
```

(Perl will automatically prompt you for any necessary config info, then go away and download and install **POE** and all the modules it depends on, then **XML::Simple** and its dependencies. Note that **XML::Parser** relies on the *Expat* XML parser kit – you may need to install this off your Linux distro.)

Now grab and unpack the latest *InterMezzo* sources:

```
cd /tmp
wget ftp://ftp.inter-mezzo.org/pub/intermezzo/
current/intermezzo-1.0.6.0.tar.gz
tar xvfz intermezzo-1.0.6.0.tar.gz
cd intermezzo-1.0.6.0
```

Now to build the components, at which point the real fun begins: the source distro, as of 1.0.6.0, is broken. Instead of typing **make** you will have a smoother ride if you install the Perl modules thus:

```
cd LibRSync
perl Makefile.PL && make install
cd ./SetFS
perl Makefile.PL && make install
cd ./lento
perl Makefile.PL && make install
```

Next, install the tools:

```
cd ./utils
make install
cd ./tools
make install
```

A key assumption of this article is that you're

running Kernel 2.4. If your kernel includes *intermezzo* support, skip the next step:

```
cd ../presto24
./configure && make install
```

Note that this step relies on you having a properly configured and built kernel source tree in */usr/src/linux* – if not, you can tell '**configure**' where to find it by supplying the **--enable-linuxdir=<path>** option: for example:

```
./configure --enable-linuxdir=/usr/src/linux-2.4.12
```

You might also run into trouble if the include file *linux/version.h* doesn't have a **#define** for the symbol **UTS_RELEASE** (kernels prior to 2.4.10).

After building and installing the *Presto* server (which, somewhat confusingly, creates a kernel module called *intermezzo*, along with some device nodes called */dev/intermezzo0* to */dev/intermezzo3*), you should create a new group for *InterMezzo* users:

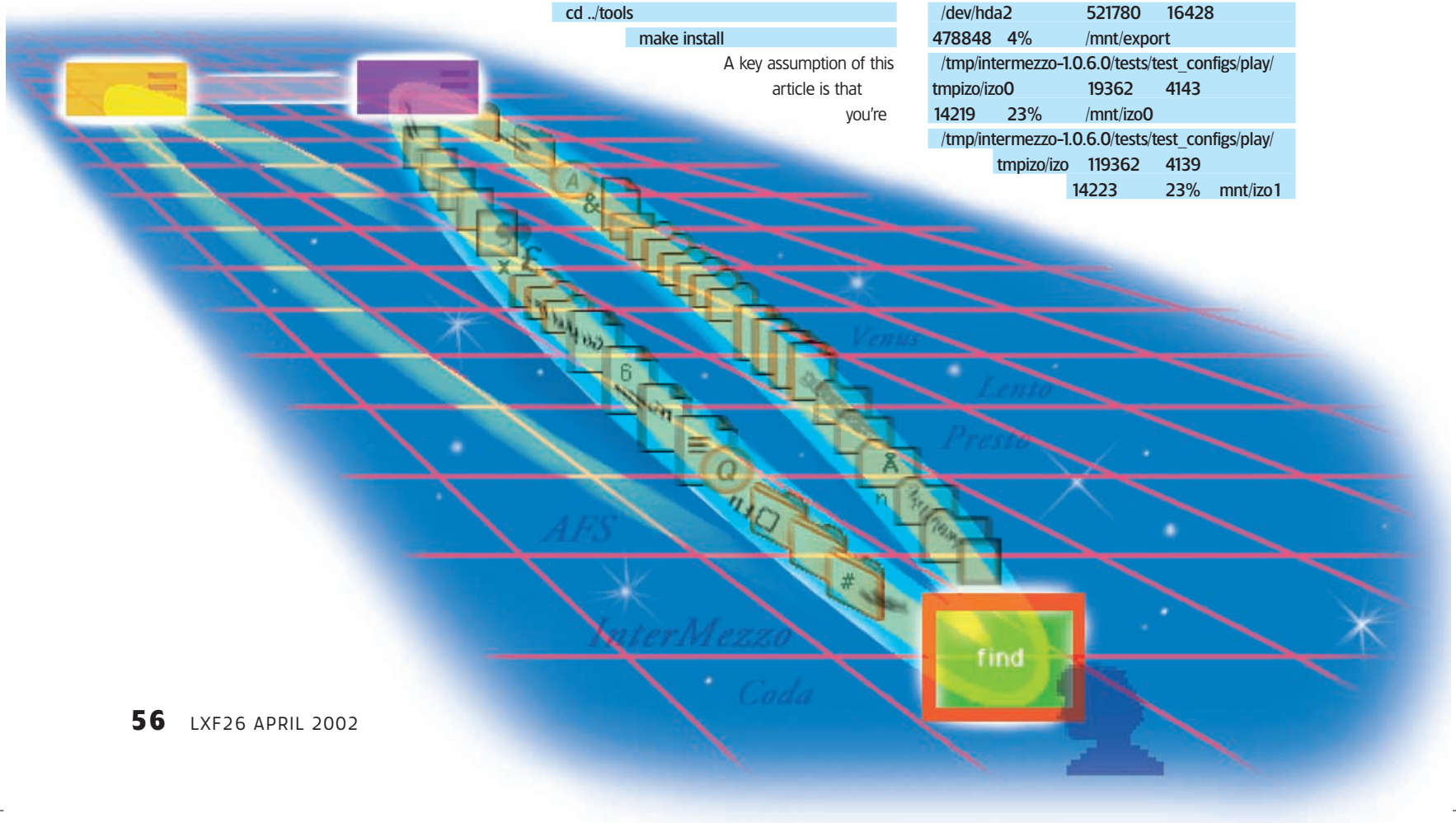
```
groupadd -g 4711 InterMezzo
```

Now can finally try the test scripts:

```
cd ../tests
make play
```

df see some rather baffling looking new filesystems:

Filesystem	1k-blocks	Used
Available	Use%	Mounted on
/dev/hda4	7834788	6307208
1129572	85%	/
/dev/sda3	3397636	2167072
1230564	64%	/mnt/stuff1
/dev/sdb1	4192800	1047348
3145452	25%	/mnt/stuff3
/dev/hda3	11258680	9170556
1516212	86%	/home
shmfs	385784	0
385784	0%	/dev/shm
/dev/hda2	521780	16428
478848	4%	/mnt/export
/tmp/intermezzo-1.0.6.0/tests/test_configs/play/		
tmpizo/izo0	19362	4143
14219	23%	/mnt/izo0
/tmp/intermezzo-1.0.6.0/tests/test_configs/play/		
tmpizo/izo	119362	4139
14223	23%	/mnt/izo1




```
/tmp/intermezzo-1.0.6.0/tests/test_configs/play
/tmp/izo2      19362  4139
14223  23%    /mnt/izo2
```

You can end the test by hitting <CTRL>D to close the *Xterm* or shell session, or if necessary by typing **make halt** to shut down the *InterMezzo* server.

Unlike the other README files in the source distribution, the one in *intermezzo-1.0.6.0/test* actually has something useful to say (other than “please read the HOWTO”, which lies); there’s a full testing framework here. You can find logfiles left behind by the tests in the subdirectory *intermezzo-1.0.6.0/tests/test_configs/play*; these can be quite informative if any errors have occurred.

» I’ve installed it, now how do I configure it?

Write config files that tell *InterMezzo* how to behave, create the filesystems, mount them, then start the *Lento* cache managers on each machine.

The config files for XML format *InterMezzo* live in the directory */etc/intermezzo*. There are three core files: a system ID file (on the server), that describes the server; a server database that lists known servers; and a fileset database that associates filesets with client machines. *E.g.*, on our desktop PC server, */etc/intermezzo/sysid*, which identifies the system to *InterMezzo*, should hold the following:

```
<sysid name="myPC" psdev="/dev/intermezzo0"
bindaddr="192.168.1.13"/>
```

On the laptop client, */etc/intermezzo/sysid* looks like:

```
<sysid name="myLaptop" psdev=
"/dev/intermezzo0" bindaddr="192.168.1.31" />
```

The server database */etc/intermezzo/serverdb*, present on the PC server, looks like this:

```
<serverdb>
<server name="myPC" ipaddr="192.168.1.13"
port="2222" bindaddr="192.168.1.13"/>
</serverdb>
```

You can leave out the **bindaddr** address, unless you’re running an *InterMezzo* client on the same machine as a server – in which case you need to bind the client to a different IP address from the server.

The fileset database */etc/intermezzo/fsetdb*, present on both machines, looks like this:

```
<fsetdb>
<fileset name="shared" servername="myPC"
fetchtype="Rsync">
<replicator>myLaptop</replicator>
</fsetdb>
```

This file declares that the fileset “shared” is served by “myPC” and replicated on the client called “myLaptop”. The *fetchtype* specifies that *InterMezzo* synchronises files using the *rsync* protocol; the built-in simple *InterMezzo* bulk mover is called “Desc”.

Where is the actual store of files set up? So far we’ve got a name – a fileset called “shared” – but no files. The answer lies in some weird magic we add to */etc/fstab*, the filesystem mount table. This needs to include an entry that looks something like this:

```
/dev/hda4 /mnt/shared intermezzo fileset=shared
,mtp=/mnt/shared,prestodev=/dev/intermezzo0,
cache_type=ext3,data=journal,noauto 0 0
```

(Note that all of this belongs on a single line!).

Here we see the entry for the desktop PC server. It declares that the partition */dev/hda4* is mounted on */mnt/shared* as type **intermezzo**. A whole load of special mount options follow it, which specify that its contents are the fileset called **shared**, that the mount point is */mnt/shared*, it is to use the *Presto* synchronization device */etc/intermezzo0*, the cache filesystem is of type *ext3*, and data is journaled. (Whew!) It’s actually just like a normal entry in */etc/fstab*, except for the huge list of special mount options that define the fileset’s name and attributes.

Having set up the configuration files, we use the *mkizofs* tool in the *InterMezzo* distribution to initialise the *InterMezzo* filesystem on */dev/hda4*:

```
mkizofs -t ext3 -r shared -j /dev/hda4
```

If you mount the filesystem as type *ext3*, you’ll see that in addition to the normal *lost+found* directory, the root of the filesystem contains a directory called *.intermezzo* that contains various other stuff. Don’t mess with this. In actual fact, you want to mount it via the **mount** command, like this:

```
insmod intermezzo # just in case
```

```
mount /mnt/shared
```

Type **df -T**, you will see something like this:

```
clueless:/etc/intermezzo # df -T
Filesystem Type 1k-blocks Used
Available Use% Mounted on
/dev/hda4 ext3 7834788 6307260
1129520 85% /
/dev/hda3 ext3 11258680 9170556
1516212 86% /home
shmfs shm 385784 0
385784 0% /dev/shm
/dev/hda2 intermezzo 521780 16452
478824 4% /mnt/shared
```

The *InterMezzo* filesystem is mounted, but you can’t read or write to it from the local machine – not without actually firing up the *Lento* cache manager, that provides access to it from *InterMezzo* clients:

```
/usr/bin/lento &
```

Now that *Lento* is running (assuming it doesn’t print lots of hostile error messages – usually because something in the config files confused it), you can go to work on the laptop client.

Assuming the *InterMezzo* software is compiled and installed as on the PC, all you need to do is copy the contents of */etc/intermezzo* from the PC to the laptop, tweak the values in */etc/intermezzo/sysid* to say that it is the laptop, and then add a line to */etc/fstab* to designate where the cache filesystem lives. You can use a loopback-mounted filesystem, *i.e.* one that is contained in a large file on disk instead of a partition, if you want. Create it like this:

```
dd if=/dev/zero of=/home/cache bs=1024
count=1000000
```

(To create a just-under-1 Gb file containing

nothing but zeros, called */home/cache*).

Then run *mkizofs*:

```
mkizofs -t ext3 -r shared -j /home/cache
```

The program will warn you that this isn’t a block special device and ask if you want to go ahead: say “yes.”

Now add a line to */etc/fstab*, like this:

```
/home/cache /mnt/shared intermezzo
fileset=shared,mtp=/mnt/shared,prestodev=
/dev/intermezzo0,cache_type=ext3,data=
journal,noauto,loop 0 0
```

This says that */home/cache* is mounted on */mnt/shared*, and is a filesystem of type **intermezzo**; note the “loop” parameter at the end, indicating that it’s a loopback device.

Finally, mount the filesystem and run *Lento* on the client:

```
mount /mnt/shared
/usr/bin/lento &
```

» So how do I use it, now it’s configured?

You use it just like any other filesystem. Drop a file into */mnt/shared* on your laptop (myLaptop) and a few seconds later it shows up in */mnt/shared* on your desktop (myPC). The same happens in the other direction, too.


This sounds just like NFS, only infinitely harder to set up. The big win comes when you disconnect the laptop. Shut down the *Lento* process on the laptop:

```
killall -HUP /usr/bin/lento
```

And carry it away. While you’re away, you can edit files in */mnt/shared* – which is still mounted. When you get home again, plug your laptop into the network and run *Lento* again, and any changes to */mnt/shared* on your laptop will be automatically propagated to the server.

» Is it safe? Is it stable?

Remember, *InterMezzo* is under active development! The *Lento* cache manager is still a prototype written in Perl. There’s no encryption built into the protocol for client/server communication – if you’re using a wireless network, you would be well advised to set up a TCP/IP tunnel on port 2222 using *ssh* (which goes some way beyond the scope of this article!). And we managed to lock up our test laptop while writing this article by physically ejecting a PCMCIA network card while *Lento* was trying to send an update to the file server.

On the other hand, *InterMezzo* uses *ext3* for storing files: and this is nearly ready for mission-critical applications. So files written locally are probably as safe as houses; the only major problem is likely to be failure to replicate changes to another machine. The *Presto* module, which lies on top of the *ext3* layer and handles file i/o, is now part of the 2.4 kernel source tree and can also be considered stable: only the user-space side of the system (which handles client/server data synchronization) is a bit rough. 

Emulators



Arcade heaven

Simon Goodwin plays with XMAME, the arcade machine emulator for Linux.



Your coverdisc includes emulator sources and many of the utilities mentioned in the article.

This month we test *MAME*, the *Multiple Arcade Machine Emulator*. This ambitious project emulates coin-operated arcade games, rather than home consoles or general-purpose computers. It's not one emulator, but a collection of modules emulating arcade cabinets custom-made to run thousands of microprocessor-based arcade games.

MAME is just five years old, though its originator, Nicola Salmoria, had a solid reputation writing Amiga freeware a decade ago. More than 100 programmers have bolstered his effort, and *MAME* has eclipsed earlier multi-emulation efforts like *Laser*, *Replay* and *System 16 (LXF19)*, which supported specific families of arcade game hardware.

MAME shares processor emulation code with computer emulators based around the same chips, augmented by routines to mimic hardware added specifically for arcade use. *MAME*'s modular design takes advantage of the hardware shared between products, developers and manufacturers. Source for *XMAME*, the Unix version, is on our coverdisc, with binary packages for common Linux systems. *XMAME* is one of the biggest emulators around, totalling 40MB of source, plus the game ROMs needed to bring it to life. The source includes driver data for over 3000 different ROM sets, dating from 1975 to 2001.

Electronic arcades evolved from Pinball, the first coin-operated entertainment with no prizes other than extended play, and custom themes and backdrops to update old cabinets. *MAME* emulates all the greatest microprocessor-based arcade games, and new ROMs are periodically added. If you've a set of Deniam, Konami GX, Sun, Orca or Seibu ROMs in the back of your shed,

the *MAME* team would like to hear from you. They are particularly seeking Sega System 16 and 24 ROMs.

The documentation totals a megabyte of text, HTML and *Latex* data. Key information about each ROM set is embedded in the 13.5MB *XMAME* executable, indicating the screen format, control keys and which chips to emulate. Tab calls up controller mappings and other useful information while a game runs.

Graphic examples

XMAME displays can use an X11 window; *SVGAlib* for Linux on IBM-based PCs; GGI or SDL; OpenGL and 3Dfx extensions for X. The *SVGAlib* version can also drive 3Dfx hardware directly.

XMame 0.57 is fussy about the bit depth of the X display. The stable Debian 0.36 package works happily in my preferred 24 bit mode, but the latest release gave a garbled display and periodic segfaults until I bodged XF86Config to specify 16 bits per pixel.

Even *Space Invaders* – basically mono with simple gel strips overlayed top and bottom – claimed to need 32770 colours, when four might actually be enough. It appears that a maintainer – possibly for the more prescriptive Windows code – has wired in a dependency which Linux users would be better off without.

Glide and OpenGL versions support bilinear filtering for neater textures, and hardware anti-aliasing for smooth vector displays. *FXmame* is the Glide 2 version for 3Dfx graphics cards. It can use gamma correction to match colours and intensities more closely, but only runs as root as the driver needs direct access to 3Dfx chips which only the superuser can gain.

GLmame is the OpenGL version, with portable 3D support

through the *OpenGL* abstract interface to 3D hardware and *libjpeg* compression. It can even emulate phosphor trails across the screen of vector displays, though not the flicker as they neared their redraw limit. It works with MESA 3, even in software-only mode, though MESA 4 is more reliable and you really need 3D graphics hardware such as ATI, Matrox or NVidia for a playable frame-rate. *XFree86 version 4* and 16 bit colour are needed.

Unconstrained by Microsoft, arcade designers pick the ideal processors for each product, tracking the best in CPU architectures. *MAME* emulates many once-trendy chips, including Motorola's 6800, 6802, 6805 and 6809, Intel's 8038, 8039, 8041 and 8085, and Texas Instrument's CPUs and DSPs like the 9900, 320 and 340 series. Chuck Peddle's ubiquitous 6502 and Zilog's Z80, Z180 and Z8000 are also much used. *MAME* can emulate MIPS RISC processors like those used in recent Sony and Sega consoles, and obscurer parts like the 2100 DSP, a 6809 variant from Konami, the 6309 – a souped up 6809 with extra goodies like a DIV instruction – and NEC's V series, first based on 16 bit Intel x86's, later embellished as the V60.

CCPU – Seventies RISC

CCPU is not a CPU optimised for the C language, though the PDP-11 minicomputer, emulated here by *T11*, comes pretty close. *CCPU* emulates a ground-breaking 1978-vintage 12 bit RISC specially invented for a dozen Cinematronics vector games like *Ripoff*, *Tail Gunner*, *Spacewar* and *Star Castle*. Another game special is *ASAP*, the Atari Simplified Architecture Processor, a custom RISC used in the *BeatHead* game.

Interchangeable graphics, sound and processor emulation modules do much of the work, but there are specific machine, driver and video hardware code modules for many of the ROM sets. These glue together the generic component support with *ad hoc* code and hardware mapping information, reflecting the custom wiring in each production cabinet.

Audio modules emulate Yamaha FM and OPL synthesis, GI's AY8910 and Atari Pokey beepers, a subset of the Maplin catalogue stalwart Philips SAA1099, generic DAC and ADPCM outputs, and dozens more, including speech chips like the TI5220 and Votrax speech synth, familiar from Gauntlet and SpeakAndSpell and the semi-human burbling in *Q-Bert*.

Apparently the coders at erstwhile pinball specialists Gottlieb gave up trying to program the Votrax to produce intelligible words and fed it random numbers to get a more impressive babble, like an adenoidal human using an alien language. *Q-Bert* graphics were inspired by M.C Escher, and the working title of the game was *Snots and Boogers!*



Sega's *Afterburner* made good use of dual 68000 processors and a Z80 sound controller.

Making MAME

MAME is well documented and available ready-compiled for many systems. *XMAME* updates lag slightly behind new Windows versions, but usually only by a few days. You may have to wait longer for PPC, ARM and 68K binaries, or versions tailored for displays other than X or *SVGAlib*.

The latest version of *XMAME* early this year was 0.57, pre-compiled in packages with the RPM update incompatible with many existing distros. Compiling from source is the best way to get code that's sure to run on your machine, tailored for your libraries and hardware, but it's a big job, especially on a slow machine; if your connection uses anything better than tins and string, a binary download is worth considering, if you can find one suitable. That might mean opting for an old version, but anything since version 0.3x will run most of the classic ROMs.

The source RPM is over 6.5 megabytes long. The tarball is packed with *bz2* compression rather than the usual *gzip*, so the name is *xmame-0.57.1.tar.bz2* rather than *.tar.gz* – but the difference is well worth it, as the *bz2* tarball weighs in at about five and a third megs. Unpack it with **tar -xvzf**; the **j** takes the place of the **z** that would signal *gzip* uncompression.

The Makefile explicitly supports DEC Alpha, Motorola 68K, PPC, Sparc, HP-PA and MIPS RISC chips as well as Intel 32 and 64 bit processors. *XMAME* has few dependencies – it expects *GCC* but you won't get far on Linux without that, and although it depends on the *zlib* compression library you don't absolutely have to install it, as a stripped-down version is included in the *XMAME* archive – uncomment the line **ZLIB=1** in the Makefile to use that.

Other options are worth tweaking in the well-commented makefile. Unless your copy of *GCC* is buggy you should use the fourth set of **CFLAGS** on an x86, commenting out the rather unambitious first set, or the fifth set for Linux on a PPC Mac or Amiga. Before you make changes type **cp makefile.unix Makefile** so you can edit a copy and simply type **make** to build your tailored version.

Set an alternative **DISPLAY METHOD** if the slow but portable X11 default is not to your liking. You must compile versions separately if you want more than one, but can specify the alternate display by tacking a parameter onto the make command, so **make DISPLAY_METHOD=svgalib** builds the metal-bashing PC variant.

The **MASM_DEBUG** symbol requests *MAME*'s built-in debuggers, but these retard emulation so consider if you really need them, typically to hack a game or implement a new ROM set. **MAME_NET** selects experimental multi-player support code, using UDP messages to sync machines across a network.

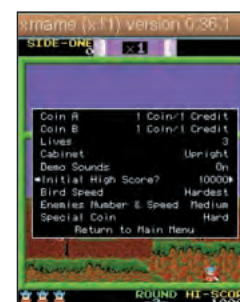
After saving the makefile, type **make** and fix a snack. My Debian 2.2 Linux took just over 40 minutes to compile the default X11 version with recommended optimisations on a K6/2-500. It munched through more than 700 C source files, without a single warning message from *GCC*.

Installation

make install copies the compiled code and documentation into the preset directories. Unless you alter those you'll need root superuser privileges to do this, as by default the files go in system directories to which a normal user does not have write access. Copy the supplied *xmame.rc.dist* file from the /doc directory to a new directory *~/xmame*, and edit it to suit



Connect up your ride-on lawnmower and an old PC and *voilà!*



This configuration menu lets *MAME* users access switches normally hidden inside a *BombJack* cabinet.



Atari's vector scan *Asteroids* delivered far higher resolution than traditional raster-based games.



Emulators

◀ yourself. Most of the files for specific games go in subdirectories of /usr, where they can readily be shared.

The ROM files are copyright and their usage is strictly controlled for legal reasons. The canonical site, 'www.mame.dk', gives legal justifications for their archive, including backups, education and emulator programming. There are no ROMs on our cover discs, but plenty online; *LXF19* included tips on buying original ROM sets.

Filenames and letter case must be just right or *XMAME* will not start a set of ROMs. Remember to check the case of the sub-directory name as well as the ROM; prefix the launch command with **strace** to check the exact paths if necessary. Case dependence is a Unix quirk irrelevant to other *MAME* platforms, so renaming is often necessary.

The mame.dk website includes notes on each ROM set, typically covering gameplay tips, variants, bugs and other history. The only essential command line parameter is the ROM set name – though it defaults to the name set in the config file, typically *pacman* – but almost 200 lines of help are available if you specify the **-?** switch as an argument. Many **-list** options give details of the supported game ROMs and the chips they use. The **-listcpu** variant gives a wide chart of CPU popularity by year; with the Z80, 6809 and 68000 predominating a varied list.



Taito's *Space Invaders* triggered a small change shortage in Japan.

MAME may put ROMs and other shared files in /usr/lib/games/xmame or /usr/games/lib/xmame, depending on the package you have installed. You can change the path by editing xmamerc, but the FAQ is misleading for version 0.57 running under X11; the file needs to be called xmame-x11rc, not xmamerc as used for older versions, and the directives it holds should be specifically for that version, not for other types of display – it will ignore ones for, say DGA, warning if it finds any.

Initial Options

The option **-?** calls up hundreds of lines of documentation about *MAME*'s other command line options. You can preset these in a configuration file for each user or ROM on your system, or save typing by running a front-end program. Many front-ends for *MAME* are available; for instance *MAMECAT* uses Trolltech's Qt graphics toolkit and adds graphic previews and game search options, whereas *gRustibus* is tailored for the *GNOME* environment. Portable ones like *it-mame* and *tkmame* are written in Tcl/Tk, and *kmamerun* and *gnomame* are alternatives for *KDE* and *GTK* respectively.

However invoked, useful options include **-scale** to resize the screen. Most games run at around TV-style 300 by 200 pixel low resolution, so they're easily lost on a typical X desktop. A scale

MAME games



Know thy enemy in Williams's *Defender*.



Terminator 2 lives to die again - he always said he voz going to be back...

All of the games that we tested were playable. Performance depends on the ROMs and your Linux system; older games are limited by host graphics speed, especially in X, but emulation of the multiple processors in later cabinets puts more strain on your CPU. *MAME* interprets all game code; 68K Linux might run the code for some games directly, but *MAME* opts to emulate them all, for consistency and portability, favouring x86 or RISC platforms.

Pacman

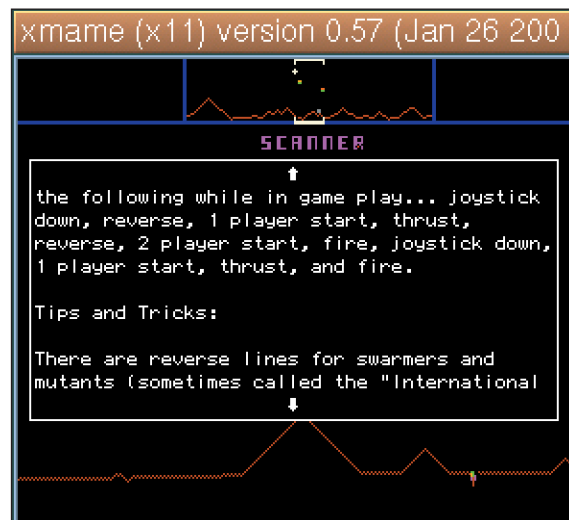
Eighties arcade hits are often playable even on a slowish Linux box. The original *Pacman* ROM set is called *Puckman*, matching the name it was given, and the main character's resemblance to a hockey puck, till the distributors panicked that vandals would find it easy to change the initial letter into an 'F'. Its humble 3 MHz Z80 delivered less CPU power than a ZX-80, bolstered by Namco sound and 16 colour sprite graphics with 288 by 224 pixels at 60 Hertz, normally orientated vertically rather than horizontally. Sega's *Pengo* was essentially a 32 colour version.

MAME's 'DIP Switches' menu offers an alternative 'Cocktail' tabletop display format as well as *Pacman*'s upright default, with Normal or Hard difficulty and an alternate set of ghost names more intelligible to an English audience. Initial number of lives per coin can be set between one and five, with bonus lives available every 10,000, 15,000 or 20,000 points, or none if you're running an arcade where you want to discourage expert players.

Service mode lets you test the key assignments, playing a different sound every time it detects a button-press, joystick move, or coin entry. This was originally intended for engineers checking out the cabinet, but is all part of the authentic *MAME* experience and may be handy if you're building a replica cabinet and want to test controls patched over the computer keyboard.

Eight bit eighties

Defender runs on a 1 MHz 6809 with a slightly slower 6808 driving Sam Decker's awesome DAC effects. After the

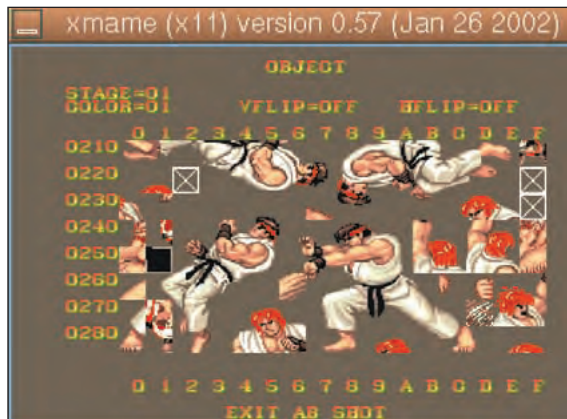


XMAME overlays tips and trick information onto the *Defender* screen.

disconcerting startup pattern – while the ROMs are checked – press F2, the internal 'advance' button, to look through the setup options and press 7, mapped to the internal button to clear the high score table, to tweak settings. *Defender*'s Game History section, on the Tab menu, has enlightening details of bugs, special tactics, and an 'easter egg' triggered if you do just the right sequence of pressing the five buttons plus joystick moves.

Taito's *Qix* runs on dual 6809s with another Motorola chip, the 6802, delivering sound by direct writes to a 'Digital to Analogue Conversion' DAC much like Linux's '/dev/dsp'. *Qix* features few colours and no scrolling, but *MAME* still has to emulate three synchronised processors to run this exceptionally original game.

The 1980-vintage Atari *Centipede* arcade game runs on a 1.5 MHz 6502 with an Atari Pokey chip, which *MAME* renders



Street Fighter 2 has a built-in editor for the sprite graphics.

factor more than two or three slows X down noticeably, and the chunky pixels become obvious, but you can work around this by using a more appropriate screen mode or a custom screen.

My 500MHz K6 ran *Afterburner 2* on a 16 bit XFree86 3.3.5 desktop at a playable 20 frames per second, four times the original size via **-scale 2**. To get a full-screen display at the original 60 Hertz I'd need to bypass X. F11 toggles a frame rate

counter at the top of the screen, and F8 and F9 adjust the frame skip, trading CPU time for update speed. **-effect 1** uses smoothing as well as double width and height, for still frames even better than the original machine, but updated at a slightly flickery 12 frames per second.

Other **-effect** options mimic the original screen scanning grid or smooth out the zoomed display. Special options for vector displays offer adjustable simulated beam size, flicker and anti-aliasing. **-ror** and **-rol** rotate the display 90 degrees each way, giving accurate portrait-mode displays as many arcade machine screens appear taller than they are wide. **-flipx** and **-flipy** do the same for mirrored displays – less useful unless you play on an autocue machine, or dangling from the rafters.

-samplefreq sets the audio output rate, making a similar trade off for samples, with muffled sound but less CPU load at the low end of this range; use 48000 or configure this for the best quality your machine can deliver. The default is half CD rate, 22050 Hertz. You can add buffers, incurring some lag but smoothing out glitches, with **-bufsize** which sets the number of buffers rather than their size. Some games insist on sound emulation, as they use it for synchronisation, but **-fakesound** allows those to run silently. **-volume** sets audio attenuation, from **-32** up to **0**, the loudest.



MAME delivers Pacman - or Puckman - pixel perfect.

rather erratically unless you finely tune the sound settings. *BombJack* runs on two Z80s (one for sound) and uses a AY8910 sound chip – like a ZX Spectrum 128.

3D vectors

Atari and Cinematronics vector games presaged modern 3D gaming with analogue wireframe hardware. Atari's rock-breaking *Asteroids* ran on a smooth vector scan display with an 8 bit CPU feeding it co-ordinates at a resolution of 1024 by 768 pixels which raster games could not match, though the refresh rate depended on the number of vectors – flickering as the screen filled up. *Asteroids* used the same physics and control system as *Space War*, developed at MIT in 1961, which begat Atari founder Nolan Bushnell's *Computer Space* a decade later.

Tempest, originally conceived as 'first person *Space Invaders*', tweaked Atari's vector hardware to work in colour, but overheating problems mean emulation is your best chance of a playable game, these days. It ran on a 6502 at 1.5 MHz with two Pokey chips, as used in the Atari 400, giving clicky MAME sound, as for *Centipede*. Mouse emulation is a nice idea but the original used a trackball and direction reversals take some getting used to, but the keys work pretty well, too.

Graphic progress

Konami's *Vulcan Venture* side-scrolling shooter uses 1024 colours and almost 3MB of ROM, including sampled speech. This is a good one to run full screen as it used a landscape display rather than the portrait layout common in arcades but not computer screens other than Radius Pivots.

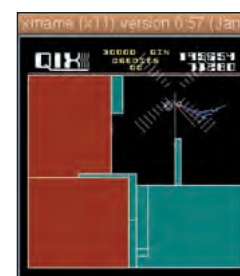
Capcom's *Street Fighter II* uses a 12 MHz 68000 with a 4 MHz Z80 driving OKI and Yamaha sound chips. The screen resolution is a wide-screen 384 by 224 pixels. It starts up with extensive self-test facilities, including a screen just to try out the sound effects.

The back tick character, top left on many PC keyboards, calls MAME-specific options, trimming the overall volume and relative

levels of stereo from the YM2151 synth and the companion OKI6295 sample player's four voices. Key and controller signals get sent to the game menu as well as the overlaid MAME one, so the arrow keys affect both displays and settings. Work around this by setting MAME ones first, then press <Esc> to concentrate on the game ROM configuration later. These MAME options also control image brightness and gamma, the ratios between preset and rendered brightness levels. You'd need a screwdriver or soldering iron to adjust these in a real arcade cabinet.

Terminator 2 runs to more than 7MB of ROM, with a 32 bit Texas TMS34010 RISC CPU plus a 2 MHz 6809 for audio, driving a DAC and two more audio chips. The 400x256 pixel display uses a 12 bit colour palette. Hidden features of this game reward exploration when you don't have to keep feeding it with coins. It plays chilling samples during its setup mode, and its attract mode sounds and graphics are quite exceptional.

Hidden menus built into a *Street Fighter 2* cabinet allow graphics editing using the underlying blocks, with vertical and horizontal flip and palette controls, configuration of the text messages, and enough other goodies to virtually constitute a second game, allowing you to make your own *Street Fighter 2 SE*, if not *Street Fighter 3*, by patient experiment. MAME has exceptional depth; if you like emulation it merits investigation, even if you've never set foot in an arcade. **LXF**



Qix is one of the most original abstract arcade games.

MAME Links

The many flavours of Mame

All formats MAME:	http://kemuator.sourceforge.net
http://www.mame.net	TKMame front end:
Arcade ROMs:	http://tkmame.retrogames.com
http://www.mame.dk	XMame home:
Grustibus front end:	http://x.mame.net
http://grustibus.sourceforge.net	XMame patches:
Kemulator front end:	http://www.win.tue.nl/~stijn/xmame

Linux Format Interview JEREMY ALLISON

Samba
king

**Richard Smedley
meets the man who
co-engineered the
Unix glue which holds
together most large
Windows networks.**

Samba is a print and file sharing technology which has become an essential part of any Windows/Linux network setup. Jeremy Allison has been hacking the Samba code since the earliest days. His cv reads like a Who's Who of Unix companies and he has been a keen advocate of Free Software, copyleft and the GNU system throughout his career.

Time has not dimmed his enthusiasm, and he recently flew back to his Sheffield homeland to speak about Samba and the benefits of Free Software to the assembled business and IT audience at the Sheffield Linux seminar, where we caught up with him.

LINUX FORMAT How did first you get involved with Samba

JEREMY 'R' ALLISON Well, Andrew [Tridgell] originally started Samba in 1991 – about the same time that Linux started. (I actually remember seeing Linus's post on the Minix newsgroup – I used to hack Minix SCSI drivers on my ST). I had just ported Vantive's software to Windows NT. Back at that time everything was proprietary so there was no difference – Solaris, SunOS, NT – all the same. Windows NT (or 32) was the first Microsoft OS where you didn't have to deal with segments – a "real system!"

We were using PC-NFS – this was Sun's product, that allows Windows PCs to connect to NFS servers and I used to have to support it when I was at Sun – and I _really_ hated it, with a

passion as only someone who has to support it and fix bugs in it [can have].

I ran across a Microsoft rep at a trade show and I said "You bastards! Why did you invent a new and totally undocumented file serving format for Windows. Why didn't you use NFS?" He said "It's not undocumented. It's X/Open spec 2.07." So, because I always buy documentation – I buy a lot of books – I bought the documentation for X/Open. I thought "This looks kind of interesting." So I started writing some code (in C++, as all good code should be) to actually implement this and then Andrew's code turned up, and it was in C, and it was a lot further advanced than mine and I thought "Oh, I'll just fix this." Ever since then I've been trying to persuade him to use C++ but I'm not preaching to the converted, I'm afraid.

LXF Once you started working on Samba, you were putting that out as GPL – were you able to do this because you, rather than the company, had the copyright?

JRA No, I didn't tell them [laughs] – I was doing it in my spare time, and I just didn't tell them what I was doing. No, I've always been a fan of the GPL, I've always had GPL'd code for a long time and in fact when I was at Sun I used to be the maintainer of the Sun 386i position-independent code patches for GCC.

LXF You had a GCC patch rejected by ...

JRA rejected by Stallman,

LXF ...rejected by God [This was how Allison had described it at the time.]

JRA Yeah, that was wonderful [laughs]. So I maintained them separately because I had a bunch of old Sun 386i's and I wanted to do position-independent code on it, because I wanted to build small X libraries – so I fixed that up and maintained it, kept it current with GCC for a while.

So I was really familiar with GPL. I knew it was a great way to do software – I just didn't tell Vantive. I said "Oh, let's try this as a file-sharing system because it's better than PC-NFS." They didn't care because they were saving a million PC-NFS licenses. So after a while I was spending a decent amount of my time maintaining Samba.

I started to infiltrate GPL code everywhere in Vantive so we switched over to using GCC from the native compiler on most platforms – because all the Vantive stuff was C++ and most of the native compilers were just crap and not portable. This was in the early days of C++.

We had to have all sorts of hideous hacks to make it work on all the different compilers, and I managed in the end successfully to argue "Look, if we write to G++ [GCC's C++ compiler]– then we have a common compiler suite that we can use against all of the platforms that we use." The one hold out, of course, was that we had to make it build in Visual C++.

LXF It must be nice to work in a company that listens to common sense arguments...

JRA Well it's Silicon Valley. Things are different there. If you can save them money they're interested.

LXF Working now at HP is well in line with your commitment to Open Source, are they still totally going the same way after the merger?

JRA I hope so. I don't know anything about the merger. As far as the merger's concerned, it's so far above my head, that you know... It doesn't really affect me. The good thing about the merger is I'm collecting Unix vendors that I



Jeremy Allison

“I said ‘let’s try this as a file-sharing system because it’s better than PC-NFS’. They didn’t care because they were saving a million licences”

Jeremy Allison

« have worked for and the only two I'm missing are Compaq and IBM. If HP buys Compaq, I'm getting them by default – that's about it. [more laughter]

LXF Hmm, they probably won't be buying IBM.

JRA Well, probably not yet, no. We did talk to IBM when we were looking for a job after VA melted down. But IBM have a standard intellectual property agreement which is ugly and we really didn't want to sign it. And HP basically did a custom IP agreement.

LXF IBM's in a really weird position with relation to Free Software. They put a billion a year into Linux, but they get it back when they sell a few..

JRA mainframes, yeah...

LXF and yet they make a billion a year from software patents.

JRA Yeah. Right now they're friendly (and I hope they remain so). Mainly because I think they're using Linux to get revenge on Microsoft. Well [laughing] from what I can see..

I've never met any business that has done a deal with Microsoft that has come out ahead. Not one. I mean there just isn't any. Nobody who's gone into any licensing agreement with them at the end of it has come out ahead. I just don't understand why people do it. It's really simple – don't sign anything, don't deal with them.

LXF Obviously (apart from Cygnus) your career path has tied in with Samba. You've been doing what you're good at, but what you wanted to do..

JRA When I was leaving SGI, someone said "shame you're leaving" and Herb [Lewis] said "same job, different office"

LXF Have you found that what companies want from Samba is always the best for developing Samba.

JRA Yeah, people do want to do weird things. We don't accept anything that is a bad idea, basically. But most things that people do are moving it forward – there are very few patches out there that are just weird, ugly stuff.

I think that the only one that we've completely and totally rejected was the ability to convert carriage return/line feed, when you copy from a Samba server.

So when we added the VFS layer in Samba – whenever anyone wanted to do that we said "Oh, that's a VFS module" and then they go away and we never hear from them again. Because it's really hard to do that. [laughs]

LXF What's really happening with the TNG [The Next Generation – a split from the main Samba project]?

JRA The TNG idea was what they wanted to do was split Samba up into multiple different daemons and have one daemon for each particular functionality. We were moving in that direction – but what we're doing is splitting Samba up into shared libraries, rather than independent daemons, because the independent daemon idea is clean, elegant and dog-slow.

When there's that level of fundamental tension in terms of how

you want to develop things the best thing to do is to say – look, it's GPL code, if you want to do this, go ahead and do this, but most of the people who ship commercial products – in fact everyone I know who ships commercial products – ships from the mainline branch, mainly because there's just more uses of it, and we beat it up more, and there's not a lot of people doing a lot of testing and a lot of development.

We now have HP, SGI, Veritas, IBM, everybody using the mainline code. Sun, I think, are going to be shipping it with Solaris 9.

LXF Microsoft use Samba a lot for testing?

JRA And internally. There's someone right now in Germany looking at a 55 thousand simultaneous user Samba system. NT just doesn't do that, 2000 doesn't do that, XP doesn't do that. You know the main problem with that system is that they want to integrate it into an existing NT domain structure – if you try and do 55k simultaneous log ons you'll melt the PDC. It just doesn't work. So

A licensing weapon "I've always been a fan of the GPL"

LXF Your career path has actively involved Samba...

JRA Funny enough the one company I wasn't really involved in Samba development, was when I was with Cygnus – I went to the CEO of Vantive and said "We should open source Vantive software – we should GPL it" and he said "how do we make money on it?" and I said "Look – right now you have to pay \$100,000 in consulting fees to set it up. The license fees that we get on top of it are irrelevant. Why don't we just give the software away, as GPL, and then charge double the prices for the consulting?" Because they were already at the time producing Vantive-licensed consultants. You seed the market with the software. People want to have very small systems they can set up themselves – good luck to them. As soon as they want to customise it they're going to come begging for consultancy help.

He said "well it's an interesting idea, we'll think about it." But after that I left because I really wanted to

work on just Free Software. So I went to work for Cygnus, working for Mike Tieman, who's now CTO of Red Hat. That was a lot of fun. I learned a lot from Michael. That's why I'm a big Red Hat fan – simply because Cygnus bought them. (I know people say "Red Hat bought Cygnus" but if you look at who ended up in charge, Cygnus bought Red Hat.)

Cygnus had the best engineering talent that I have ever worked with, bar none, and I learned a great deal from those guys. I was working on Kerberos and we ended up with a Kerberos product that would work on top of CygWin 32.

Our major competitor (Cybersource) didn't have a Windows Kerberos and so Tieman was really worried since Cygwin32 was under a completely open Free BSD licence at the time, that they would take that and use it to port their code. This was where we kind of worked out how to use the GPL as a licensing weapon. So I said "I want it to keep on being Free Software. We own all the



copyright on it. Why don't we make CygWin 32 GPL code? – not LGPL. "If we make this GPL what we can do is – if people want to make proprietary software they have to come beg us, but we can choose whom we allow to give that buyout for. If Cybersource come and ask us we just say 'no', so they won't use it because they'll then have to GPL their software." So that was actually the genesis of why CygWin 32 was done under the GPL.

And I actually used the same argument fairly successfully with Tim Wilkinson of TransVirtual technologies, who had Kaffe. He was getting all his money from the

embedded people, but he wanted to keep it Open Source so I sort of stuck him in a room and beat him up with a rubber hose for an hour or two – this was when I was working at Whistle – and I said "Look, GPL it. You own all the copyright anyway. Require copyright assignments, that way the Linux crowd get it for free and they'll keep helping you out and the people who want to embed it in a cellphone, they're going to have to pay through the nose." [smiles]

And that works really well – not to point out that I'm a man of one idea, but I actually used that at SGI as well, when they wanted to release XFS, and they were worrying about Sun.

Sun's filesystem (the UFS) just sucks, that's why they have to sell Veritas along with it. SGI was terrified that [Sun] would take XFS (that SGI wanted to put in Linux) and use it in Solaris – SGI would be giving its crown jewels to Sun. So that's why they GPL'd it. Then Sun would never be able to look at it, because they aren't let it near the core of Solaris.

they're having to do all sorts of things to move the authentication sub-systems around so that they can actually have it work on the hardware, Samba will handle that many users (touch wood). This is actually the biggest rollout that I know about, but there are smaller ones with 10-20 thousand users; but the authentication systems are the problem, because you do that many simultaneous connections to your Windows Domain Controllers and they melt, they simply stop responding, or they reboot – Windows just doesn't scale up well.

LXF You wanted to say a bit about .NET?

JRA Oh, so, Miguel and his .Net thing. Great idea...I mean. I like Miguel a lot, we get on. But I think he's very misguided – simply because what he's doing is he's putting himself on the same treadmill that we've been on for ten years – [with emphasis] and it's not pleasant. It's not fun.

Yes, they'll get the virtual machine fairly easily, but then what will happen is they will be on an endless series of



“I would much rather see the Open Source community get behind Java as a standard, not the .NET stuff”

implementing new interfaces that come out with every Service Pack. Every single service pack something new will turn up – and it won't break the old functionality, it just adds new stuff. What will happen is that they'll find that it'll use the old stuff in ways that it didn't previously do – and because Microsoft don't regression test with anything but their own stuff then what will happen is even interfaces that they had working – things will change in the way that clients access them.

So, because Microsoft will own all the clients for .NET – all the client functionality – what the Ximian folks and the Mono folks are doing is they're putting themselves on a treadmill, the speed of which Microsoft is controlling. Because it's the same treadmill that we're on. And Right now there is an alternative. There's the Java enterprise stuff – the Java beans stuff – just for me I would much rather see the Free Software

and Open Source communities get firmly behind that as a standard, not the .NET stuff.

LXF Would you say Microsoft are themselves pushing people into Free Software with the XP licence?

JRA Oh yeah. That was the best thing that they could have done. That was wonderful. They should probably ship XP with a “click here to upgrade to Linux”:

“Yes I agree to this licence.

No, I don't agree – please install Linux.” [laughter]

Which is great, actually. And the other wonderful thing that they're doing is that they're cracking down on software piracy – I love that – every time the BSA audits someone I just love it. Someone can call them up the next day and say “How about putting Linux on your desktop?”

I think over the next 2 years you'll see a significant movement.

LXF For businesses, it's also about control – you don't have people downloading stuff..

JRA No viruses..

Of course the viruses that we have now are trivial. If someone with a *brain* started writing viruses it could be really dangerous. Most viruses are script kiddie work, and they're crap. But if somebody who had real malicious intent was writing viruses

then there really would be some serious trouble.

LXF You don't have the time for that one?

JRA [laughs].. I was talking to my friend Peter Bromley about it and it's basically because all the people who have the capability to be really evil are already too gainfully employed to find the time [laughter]



The Linux stealth weapon Bringing Free Software to Windows networks

LXF Do you think Samba has been responsible for introducing Open Source into places?

JRA ... Eric Raymond once called Samba the ‘Linux stealth weapon’. The funny thing is at all the marketing presentations they always talk about Apache, Sendmail and whatever. Very few of them list Samba, even though Samba's probably just as widely used – and the reason for that is that Sendmail and Apache are externally detectable from outside a firewall, because they generally provide external services to the ‘net.

No one with any brain puts

Samba, or any SMB server, outside the firewall [laughs]. So you can't scan, and you don't know how many of them are out there.

When I was giving a talk in the US, someone came up and he confessed that he got Samba into his organisation and he'd done so by putting it on some unused machine somewhere, Linux and Samba, and he was finally found out when the NT admins kept coming to him, and asking him how they could make their NT servers as stable as his was [laughs] and what he did differently from them [more laughter] just delightful ...

Jeremy Allison

« LXF Turning now to Samba 3.0 – 2.2's been fantastically successful, the ACLs..

JRA It's a much better domain controller. The ACLs are really important. We're trying to close down the 2.2 code stream – we'll probably do at least one more release, because there's a few more nagging bugs that we know about and want to fix. Then try and get 3.0 out the door.

3.0 has the features that will reassure people that we're on the ball – “Yes, we do know about Active directory. Yes we're going to support it. No, you don't need to worry. No the Kerberos stuff isn't screwing us over!”

Andrew Bartlett's done a great job of reorganising the authentication systems. It's really funny because when he was fixing all that code, he would keep coming up with questions to Andrew [Tridge] and I – and we're looking at it going “God! Yeah, it really is kind of a mess that's grown over ten years, hasn't it? If you could make it work right, though. Make it clear and consistent, that would be great, Andrew” and he's done a wonderful job of that. He really has.

LXF So a lot of 3.0 has been internal rewrites?

JRA Internal reorganisations to make it consistent and easy to maintain. Pluggable authentication module; Pluggable back end authentication modules; making the code more modular. When Andrew and I were in Australia we took a look at server 1.0.1 (which is some tiny number of lines of code) and we actually recognised some of the function names and some of the variable names that are still in Samba today. And it's like: “We could really do with rewriting a chunk of this stuff”

Then Andrew's taken care of a lot of the Active Directory stuff, the Unicode – I finally got Andrew interested in internationalisation. With 2.2 it's internationalised to the effect that you can use it with English and one other language simultaneously. So you can have English and Japanese; English and Big 5 Chinese; or Korean, Hang or whatever.. But you can't have multiple languages simultaneously.

3.0 is fully unicode all the way through, and on the wire. Andrew had to write his own unicode package because a lot of the ones out there don't actually work right. But the main thing is the Active Directory and

Kerberos implementations, so that using Linux tools you can just add the Samba server into the Active directory back end. It works just like a Windows 2000 member server, uses Kerberos, single sign on.

We're making that easier to use – at the moment it just works. Active Directory and Kerberos are the main functionality changes. The rest of it is a lot of code tidy up; a lot of work to make things more modular.

LXF But essentially Samba's a mature product. It's 'there'..

JRA Interestingly enough there're a couple of things that're happening with Samba. One is that HP wrote the Unix extensions to Samba, which I've now (finally) put in 2.2.3 – first appearance. What they're designed to do is to do Unix to Unix file sharing over SMB. You can get full Unix details like the number of links; the device and i-node number; who owns a file; UID; GID; without having to go through a translation layer. The SMBFS maintainer in Linux has jumped on this immediately, because it makes his job a lot easier if he knows he's talking to a Samba server, rather than having to go through a translation layer to get to Unix semantics, he can just use the Unix extensions.

The next stage for the Unix extensions is to put in Posix ACLs over between server and client. Now many different Unixes have got a variant of Posix Access Control Lists in them – but none of them have a portable way of changing them over the wire. All the variants of NFS do it differently. So you can't change Sun Posix ACLs using an SGI client, or vice versa.

With all the main vendors shipping Samba, we have a portable way of modifying ACLs on a Unix filesystem – we're actually starting to extend the protocol in ways that Microsoft probably don't particularly like...

LXF So the Samba team can in some ways control the destiny of the protocol...

JRA SMB is a crap protocol, I'll be the first to admit, but at least it's one that we have a great deal of influence over. So that will be a very interesting over the next few years.

The secret is clients. The more clients we have, the more control you have over the protocol. So what I think Microsoft will be doing right now is



“We're actually starting to extend the protocol in ways that Microsoft probably don't like”

they're frantically trying to develop a new file sharing protocol, which will be completely undocumented and new.

That will be interesting because the time it takes for them to do that – how much of the (business) desktop market will they own?

If they still own 90% – yes, they're going to do it and they're going to hurt us and it's going to be very hard. Because, eventually, if everyone's on the XP licence treadmill, then they'll have to upgrade. If they only own 80%, or 70%, then it's going to be hard for them.

LXF Is a fifth of the market your target then?

JRA Once we hit 20% of the business desktop I'll be happy. Because at that point Microsoft really aren't a monopoly any more and they can't force people to do stuff. You don't have to win all of it, having a viable, non-proprietary competitor, that gets up to about 20% – at that point we

have competition again. Microsoft actually might start putting in features – trying to compete and win back some of that market share.

I'm sure you've noticed they've [recently] done a ton of development, much more than they ever did, back when they just sat there with 90% of desktops and they didn't really care. Now they've got something to fight against – they're actually working pretty hard.

LXF So you don't bear any animosity towards Microsoft employees then?

JRA At the last CIFS conference, the Microsoft engineers were helping us get roving profiles working with a Samba Primary Domain Controller. That's the level of cooperation that we have – at least at the engineering level. It's their marketing department who hate us [laughs loudly] but we actually get on with the engineers really well. **LXF**

Reviews

All the latest software and hardware reviewed and rated by our experts

LXF verdict explained

Each review is accompanied by a Linux Format Verdict to help you to assess the product at a glance (it's no substitute for actually reading the review, though). We award scores out of ten in the following categories:

Features: Does it provide the functions you need? Is it innovative?

Performance: How well does it do its job? Is it fast and reliable?

Ease-of-use: Is the interface well designed? Is the documentation well written, helpful?

Value for money: Does it have a competitive price?

For those who like numbers, the Linux Format Rating is a score out of 10 summing up the overall excellence of a product. It will usually, but need not be, an average of the above categories. We award scores as follows:

10 The close to perfect product.

8-9 Good, but has a few niggles.

6-7 Does the job, but needs work.

5-4 Average.

1-3 An utter disaster. Back to the drawing board.

The Top Stuff Award

If we really, really like something — we really think that a particular piece of software, hardware or any other sort of ware is the best stuff around — then we'll give it our Top Stuff Award. Only the very best will be chosen. It's not guaranteed to all products that score highly.



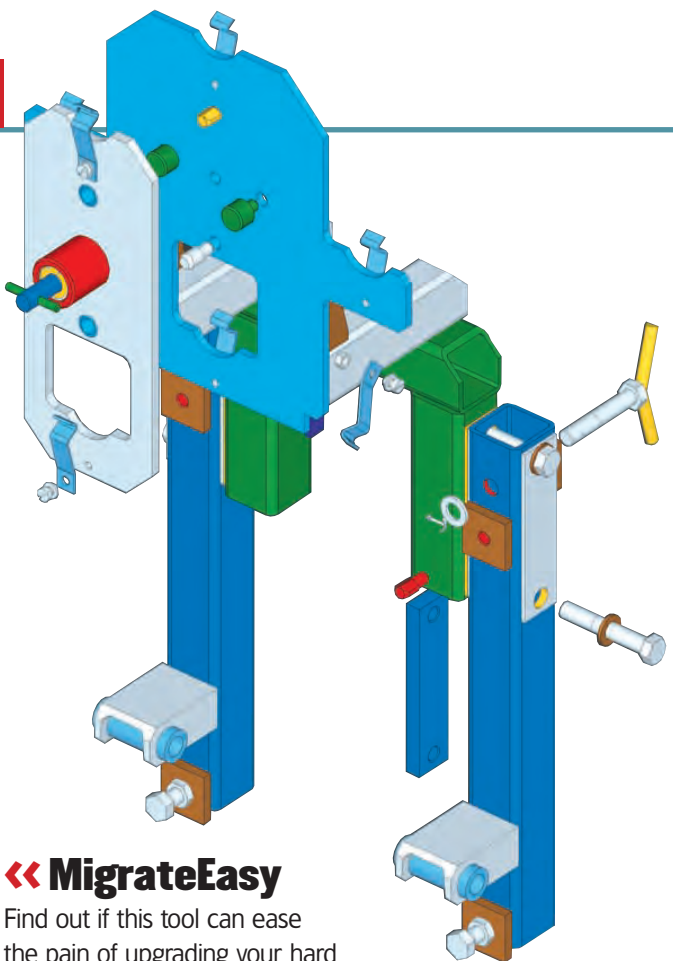
THIS MONTH...

VariCAD >>

2D and 3D design made easy with this competent engineering CAD package. What quality of package can be found beneath the quirky interface? **p28**

Sorcerer Linux

Ultimate control of your installation in this DIY distro which takes all the latest source code and builds it to your spec **p31**



<< MigrateEasy

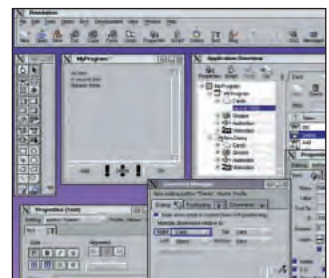
Find out if this tool can ease the pain of upgrading your hard disk. Is its convenience competitive enough? **p32**

Omis

Well established RAD suite for multi-tier application development. Does the latest point release match earlier promise? **p33**

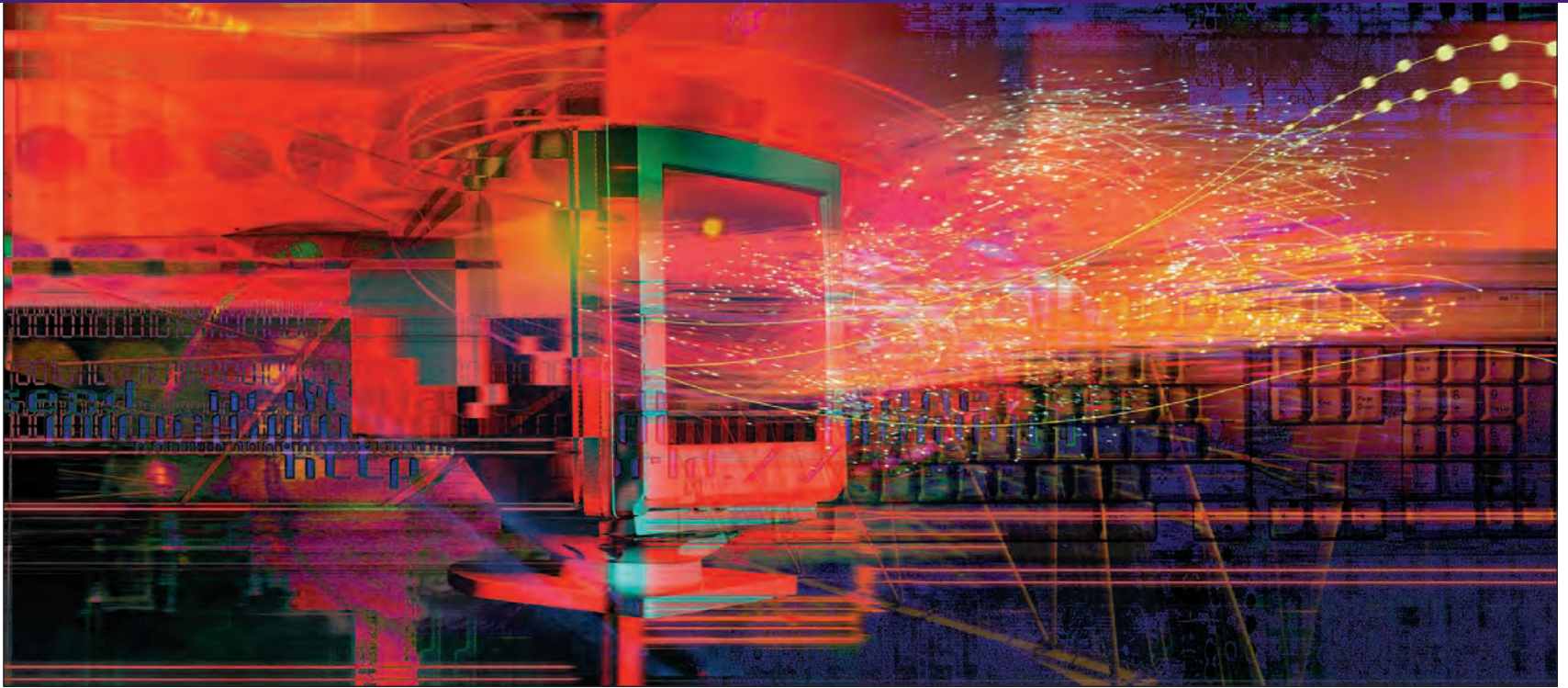
Revolution >>

The return of Apple's eighties RAD, HyperCard, updated for a cross-platform world. Can it compete with Omnis Studio and Kylix 2? **p34**



<< Books

Linux Cookbook; Zope content management; Linux in the enterprise space and eXtreme Programming with Java this month **p36**



WIRELESS ETHERNET WITH 802.11B

Wireless networking

Don't let your computer tie you down! Charlie Stross shows you how to free yourself from network cables.

Wireless ethernet is a hot new field that's sweeping the world of mobile computing. Hitherto, if you wanted to plug your laptop into the 'net you either needed a wire or a mobile 'phone – and PPP dialup over mobile 'phone was *slow*. 802.11b is a standard for radio communications between computers, running at speeds of up to 11Mbps/sec, that looks like an ethernet connection as far as the computers are concerned. (See the box *Wireless Ethernet Explained* for the grisly details, and information about related standards). In this tutorial we'll look at hooking up a Linux laptop to a private ethernet via 802.11b.

To use 802.11, you need two or more 802.11 cards, and possibly an access point. Most 802.11b cards are PCMCIA cards for laptops – the exceptions are usually PCI-bus (internal) PCMCIA adapters with a PCMCIA card already mounted! (This is because desktop PCs don't really gain much from wireless networking.) Such cards look like any other PCMCIA card, but for a small stubby aerial sticking out of the card slot. (Apple iBooks use different cards that use an aerial built into the laptop's case.)

An access point is an 802.11 base station, with antennae and an ethernet port; they serve the same purpose as an ethernet hub, allowing multiple 802.11 client systems to talk to each other and the wired-up network. Note that you only need an access point when running a wireless network in certain modes – although this will certainly be the case if you want your laptop to talk to an existing LAN or fixed internet connection. (See below for details of the modes access points can run in.)

There are drivers for a variety of 802.11 cards in the Linux kernel, as of version 2.4. The situation has been complicated a bit by some vendors who produced early cards that used competing,

non-DSSS protocols, running at different frequencies. Only a few companies actually produce chipsets for 802.11 interfaces; most of the vendors are actually resellers who re-badge the equipment. You can find a list of the different types of chipset, and the companies that use them, at http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/.

We would recommend that if you're buying an 802.11b card right now, you should try to pick one that uses the Orinoco chipset (produced by Lucent Technologies, sold by everyone, and formerly known as Wavelan). The reason for recommending this is that the ubiquitous Orinoco is well-supported under Linux – it's a bit like recommending a 3Com ethernet card, back in the good old days. You can try a non-Wavelan card if you like, but you may find it slightly harder to configure. It may not sound like the obvious choice for Linux, but an Apple AirPort makes a fine access point. The AirPort has a DHCP server, so that you can limit the 802.11b devices that access your LAN through it to known devices. It has a 10/100 ethernet cable and a modem, so that in addition to using it to bridge wired and wireless networks, you can use it as a dialup router: in this configuration, your wireless machine can send packets out to the internet via your ISP whenever you want a dialed connection. And although the configuration tool supplied by Apple runs on MacOS, there is a Java 1.2 configuration tool that runs on Linux available from <http://edge.mcs.drexel.edu/GICL/people/sevy/airport/>.

The AirPort isn't incredibly cheap (street price is around £220, although it is falling towards \$150 in the US), but if you don't mind some fiddling you can go cheaper. We tried out a NetGear ME102, which can be found for as little as £110 + VAT.

Like many access points, the ME102 is designed to be set up for the first time using a Windows utility connected via USB. Once assigned an IP address on the wired LAN, it can be managed by SNMP (simple network management protocol), but early versions of the firmware used by this device are insecure and leave it

vulnerable to a denial of service attack. While the AirPort behaves like a router, the cheaper access points are simple hubs that allow a bunch of wireless machines to talk to a wired LAN through them; they don't have DHCP servers or do dialup. (On the plus side, if you're not afraid of risking prosecution you can boost the output power on an ME102 to meltdown, see directions at: http://www.wi2600.org/mediawhore/nf0/wireless/docs/802.11/WAP11/fun_with_the_wap11.txt)

Setting up the laptop

The easiest way to get 802.11b working is to ensure that you're running a recent Linux distribution – with kernel 2.4, PCMCIA, and the *wireless-tools* package installed. The main distributions come with the kernel wireless card support compiled as loadable modules; the *wireless-tools* package provides user-space tools for configuring the cards. Because almost all wireless systems are laptops, you should ensure that your system has PCMCIA support, and then check the FAQ at http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/ to work out which kernel module you need to install to drive your card. PCMCIA support should be automatic on recent distributions such as RedHat 7.0 or later, or SuSE 7.0 or later. If you need to build a custom kernel on an older distribution, you should consult the PCMCIA-HOWTO first, as you may need to add patches or modules to your kernel. Remember, the key items you need are: 802.11b drivers for your card, PCMCIA support, and the *wireless-tools* (available in source code from http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/).

If you've got a card based on the Lucent Wavelan, try inserting it. The PCMCIA drivers should beep twice in a high tone if they've successfully loaded a recognized card. You might need

to explicitly install a kernel module to recognize the card; if so, in a root window, type:

```
insmod wlan_cs
```

(or whichever the module name for your card is – *orinoco_cs* or *wavelan2_cs* are possibilities: again, you will need to check the FAQ referenced above, to work out which kernel module to use).

If you type **lsmod** (or **/sbin/lsmod** if it isn't in your path) you will see a listing containing something like this:

Module	Size	Used by
orinoco_cs	4832	1
orinoco	27760	0 [orinoco_cs]
hermes	3328	0 [orinoco_cs orinoco]
nfsd	64880	4 (autoclean)
ds	6832	2 [orinoco_cs]
i82365	23296	2
pcmcia_core	43008	0 [orinoco_cs ds i82365]
iptable_nat	12656	0 (autoclean) (unused)
ip_conntrack	12848	1 (autoclean) [iptable_nat]
iptable_filter	1728	0 (autoclean) (unused)
ip_tables	10496	4 [iptable_nat iptable_filter]
usbcore	47264	1 [uhci]

(See how *orinoco_cs* depends on *pcmcia_core*, and *hermes* – the high-level interface to the 802.11b stack – depends on *orinoco*, which in turn sits on *orinoco_cs*.)

If you type **dmesg** to see the kernel messages, you should

Apple AirPort – shapely, fully-featured and, of course, it works with Linux.



Wireless Ethernet Explained

The tangle-free connection options

Let's start with a non-technical description: the IEEE 802.11 standard – also known as WiFi – describes a system for sending ethernet packets over Ultra High Frequency radio. Unlike traditional 802.2 wired ethernet, there are no wires. Like 802.2, it's a packet-based protocol. So basically you use it to hook networks of computers together, into LANs or WANs, without using wires.

Now let's get technical: 802.11 networks run on gigahertz-frequency wavelengths – microwave frequencies – using spread-spectrum and frequency hopping to enable a bunch of radio transceivers to talk to one another without interfering. Spread spectrum “spreads” a signal's power over a wider band of frequencies, sacrificing bandwidth in order to gain signal-to-noise performance. The spreading process makes the data signal much less susceptible to electrical noise than conventional radio modulation techniques. Frequency hopping takes the data signal and modulates it with a carrier signal that hops from frequency to frequency over a wide band of frequencies. This reduces interference because an interfering signal from a narrowband system will only affect the spread spectrum signal if both are transmitting at the same frequency at the same time. A

frequency hopping radio will hop the carrier frequency over the 2.4 GHz frequency band between 2.400 GHz and 2.483 GHz.

Traditional ethernet works by allowing any network card on the LAN to spit out a packet whenever it feels like it; if it's mangled by a collision with a packet from another computer, the protocol dictates that each machine “back off” (wait for a while) then try again. The Direct Sequence Spread Spectrum (DSSS) radios in your wireless ethernet card do more or less the same thing, varying frequencies instead of time to avoid packet collisions. The result is that a bunch of radio transmitters can all transmit in roughly the same frequency band at the same time without drowning each other out – a vital prerequisite for running a network.

The IEEE standard 802.11 specifies how DSSS data communications should work. The 802.11b sub-standard established requirements for a wireless LAN running at 11Mbps on DSSS radios using the 2.4GHz band; the newer 802.11g standard (coming later this year) should use 5.6GHz frequencies and boost the bit rate to 54Mbps. It's important to note that 802.11, even though it runs on the same 2.4GHz frequency as Bluetooth, is *not* the same as Bluetooth. Bluetooth is a

1.2Mbps wireless networking protocol that is limited to a range of about ten metres. However, before you write this off as poor, consider that Bluetooth devices are physically smaller than WiFi cards – they can be crammed into CF cards – and cost a lot less to manufacture. They also use so little power that you can run one off a PDA or cellphone without draining its batteries. (A Netgear MA402 802.11b card, in contrast, can shorten the lifespan of a laptop's battery by 30%, sucking almost as much juice as the backlight).

Bluetooth is designed for the emerging field of PANs – Personal Area Networks. A PAN might typically include your palmtop, laptop, printer, cellphone, and other detachable gadgets (such as a keyboard). Its goal is to replace cables between peripherals that don't need to communicate at high speed. 802.11b, in contrast, is designed to replace wired ethernet, and connect high-bandwidth devices at long distances. So in a year or three you might have a PAN of Bluetooth-connected devices, linked to the internet via an 802.11g enabled device.

Bluetooth is supported by Linux – IBM released their *BlueDrekar* protocol stack as Free Software in 2001 – but there are less Bluetooth devices out there at this time.

LinuxFormatTutorialWirelessLAN

« also see something like this towards the end of the listing:

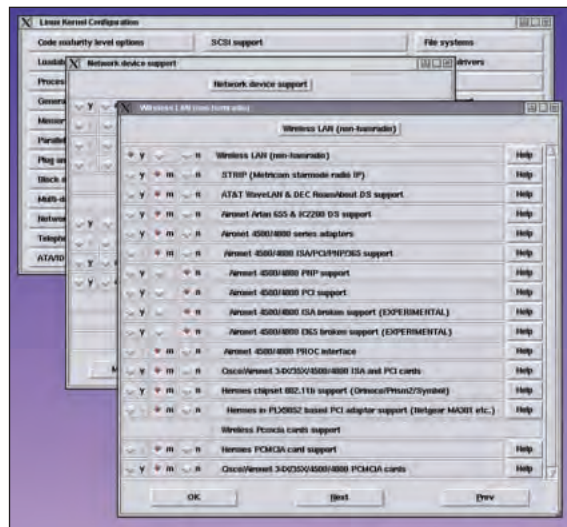
```
hermes.c: 12 Dec 2000 David Gibson
<hermes@gibson.dropbear.id.au>
orinoco.c 0.07 (David Gibson
<hermes@gibson.dropbear.id.au> and others)
dldwd: David's Less Dodgy WaveLAN/IEEE Driver
orinoco_cs.c 0.06 (David Gibson
<hermes@gibson.dropbear.id.au> and others)
eth0: Firmware ID 1F vendor 0x3 (Samsung) version 0.08
eth0: Ad-hoc demo mode supported.
eth0: MAC address 00:30:AB:0E:86:17
eth0: Station name "Prism I"
eth0: ready
eth0: index 0x01: Vcc 5.0, irq 3, io 0x0100-0x013f
```

What this tells you is that a card using the *orinoco_cs* low level driver has been detected and registered as ethernet device *eth0*, with MAC address 00:30:AB:0E:86:17.

Having an ethernet card in the system is all very well, but it doesn't do much good until you connect it to the network – which is where the fun starts. You need to first decide on a network type, then configure your access point, configure the 802.11b card to talk to the access point, then set up your routing tables. 802.11 provides for a number of different types of network – of these, the most relevant to us are Ad-Hoc and Infrastructure. An Ad-Hoc network consists of a bunch of mobile devices with 802.11 interfaces, all talking to one another; there's no access point (hub) and no connectivity outside the local Ad-Hoc network. Ad-Hoc networks can be set up by simply assigning the mobile devices a channel (that is, a set of frequencies to operate in) and a network ID (or domain). In contrast, an Infrastructure network provides a designated access point, which can forward packets from the mobile devices to an outside network (such as your wired LAN or the Internet).

WLAN cards operating on different frequencies don't communicate with each other. In addition, WLAN cards are assigned a different network ID (known in 802.11 jargon as an ESSID). Cards with different ESSIDs don't communicate, either. So you should decide on a channel and ESSID, and assign them to your access point, while keeping a note of the values for your mobile machine's card. The ESSID is a simple alphanumeric string that identifies all the people who want to be on the same network – it's not like an IP address, which must be obtained from RIPE, or an ethernet card's MAC address, which is permanently burned in.

Wireless networking is notoriously insecure; you should if at all possible set up WEP encryption, but be advised that WEP relies on RC-4 and is notoriously weak – tools (such as *airsnort* and



Recent Linux distributions – and the modular nature of the kernel – may save you the trouble of a recompile.

etherreal) exist that can break this level of encryption. If you're going to do anything significant over an 802.11 network, be prepared to use *ssh* a lot, or set up a virtual private network (VPN) on top of the wireless ethernet. That said, it's probably best to get everything working without WEP, then switch WEP on later.

Once you have set your access point to a given channel and ESSID, by whatever means come to hand (and this may mean running a Windows configuration tool to tweak your access point's configuration over a USB cable, which is outside the scope of this article!), you can turn your attention to your PCMCIA Wi-Fi card.

When the drivers are loaded, you configure your 802.11b card using the Linux wireless tools. These are included with most recent distributions, or the source code can be obtained as noted above. (Instructions for compiling them can be found in the INSTALL file in the source archive.) The wireless tools have a text-based interface; the programs are as follows:

- *iwconfig* manipulates the basic wireless networking parameters
- *iwlist* lists addresses, frequencies, bit rates and other items
- *iwspy* checks the link quality on a per-node basis
- *iwpriv* manipulates driver-specific (private) Wireless Extensions.

The most important of these programs is *iwconfig*. Just as *ifconfig* manipulates networking interfaces in the Linux kernel, *iwconfig* manipulates wireless networking devices in the kernel. And we want to bring up a wireless interface.

Suppose you've configured your access point to use the ESSID "private-wlan" on channel 1. (Essentially the ESSID acts

802.11b and the Law

Make sure that your transmissions are legal

802.11b networking runs on the 2.4GHz waveband. This is reserved for scientific, medical and industrial (SMI) applications. It isn't licensed, but there are strict legal limits on how strong transmitters may be, and there is a blanket ban on commercial use of this wavelength. In California many cafés now offer 802.11b access points with broadband internet connections, for their customers' convenience; a coffee shop offering this service in the UK would be breaking the law.

Also in California – and rapidly spreading – there's a movement for users with broadband connections to open up their access points to passers-by; see

<http://consume.net/> for details of this open internet movement (which ultimately aims to immerse us all in a sea of peer-to-peer wireless networking, making conventional ISPs obsolete). While some brave amateurs are experimenting with this in the UK, commercial use for any purpose is effectively illegal – you can use it within your office or business, but not for talking to customers or selling services, or even helping to sell a service.

However, hope is at hand: the Radiocommunications Agency (which handles spectrum allocation) has recognized demand for 2.4GHz spectrum, and for the

5.6GHz spectrum required for 802.11g. 5.6GHz is expected to be opened up for commercial applications by summer, and a public consultation on opening up 2.4GHz ended on February 15th: the review summary can be found at <http://www.radio.gov.uk/smag/24ghzinx.htm>.

Companies can now obtain experimentation licenses on a case-by-case basis (expiring after six months) to test possible commercial applications on the 2.4GHz frequency; it looks as if commercial 802.11b services may be legalised before 2002 is out.

like a call sign or network address, and channel 1 specifies the frequency the WLAN is going to run on.) You need to tell your Wavelan card (which the kernel recognizes as `eth0` – see the kernel messages earlier) to use these parameters. You can do so by running the following command as root:

```
/sbin/iwconfig eth0 essid "private-wlan" channel 1
```

Remember, if you're using a different ESSID, or a different channel number, change the values accordingly. *iwconfig* has a slew of different configurable items – type **man iwconfig** to see the man page and a list of its options. You can use *iwconfig* to set a WEP key to use, and if you have set a key on the access point and enabled WEP encryption this will be used to encrypt traffic between your card and the base station. (Note that the WEP implementation in the stock kernel modules for the *orinoco_cs* driver and the ME102 access point are sufficiently broken that this may not work! If you get into trouble setting up WEP using the standard drivers, you may want to update to a newer kernel or try downloading the Linux *WLAN-NG* experimental drivers from [ftp://ftp.linux-wlan.org/pub/linux-wlan-ng/](http://ftp.linux-wlan.org/pub/linux-wlan-ng/)).

You can also use *iwconfig* to enable power-saving modes on your card, if it's supported; this is important due to the high power consumption of most Wi-Fi cards (which is often comparable to the power drain from the backlight of a laptop's LCD display).

If you've successfully set `eth0` to the same ESSID and channel as the access point, you can now set up routing using **/sbin/route**, as you would for a normal ethernet card. For example, if you are using the TCP/IP reserved network 192.168.1, and have a router at 192.168.1.1 while your laptop is 192.168.1.2, you can do something like this:

```
/sbin/ifconfig eth0 192.168.1.2
```

```
/sbin/route add 192.168.1.1 eth0
```

```
/sbin/route add default gw 192.168.1.1
```

This sets the TCP/IP address to use on `eth0`, adds a route via `eth0` to the router, and specifies that everything goes to the router (as a gateway) by default.

Some tools are slowly becoming available on Linux that put a prettier face on all this. While the distributions don't yet have graphical interfaces for configuring Wi-Fi cards, they can be expected to arrive very soon as wireless networking takes off. In the meantime, if you use *KDE* you may want to check out *Korinoco*, from <http://korinoco.sourceforge.net/> – this is a *KDE* utility for configuring your 802.11b card. Again, if you use an Apple AirPort access point you are in luck – the java utility (referenced above) gives you a comfortable interface for configuring it.

Limitations of wireless ethernet

The zeroth limitation of 802.11 is security. Bluntly, wireless ethernet relies on radio transmissions. This means that your precious data is being transmitted where any snoop with a receiver and aerial may be able to pick it up. This is in stark contrast to conventional ethernet cables, where you have a reasonable expectation of privacy over a private LAN. (See the box, *Wireless Ethernet Security*, for what to do about this.) In conjunction with the legality issue (see the box, *802.11b and the Law*), the security problems place limits on what you can safely do with 802.11 networks. The protocol itself also imposes some restrictions on what you can do with a WLAN. The first and most important drawback is that microwaves (such as the 2.4GHz radio waves that 802.11b uses) don't go round corners or penetrate solid surfaces. This may actually be a good thing, depending on your point of view – if you are worried about drive-by hacking, knowing that there are stone walls thick enough to block the

Wireless Ethernet Security

How to not send your data to your competitors

Anyone with a DSSS card can monitor traffic on an unencrypted 802.11b network. Because it's radio, it's effectively public. In order to secure the wireless network, a standard was adopted – WEP (Wired Equivalent Privacy). WEP uses the RC4 stream cipher to encrypt packets, using a 60-bit key. Unfortunately, the first 24 bits of the key are fixed, leaving just a 36-bit space to search. Weaknesses in the cipher algorithm are described at http://www.eyetap.org/~rguerra/toronto2001/rc4_ksaproc.pdf and *airsnort*, a tool that can passively monitor a WEP-encrypted network and extract the keys after monitoring a few hundred thousand packets, can be found at <http://www.be-secure.com/airsnort.html>.

Just in case you didn't get the message loud and clear: even using WEP encryption is no

guarantee that your wireless network is secure!

Newer systems (notably Apple's AirPort 2.0 access point and some new 802.11b and 802.11g cards) use 128 bit keys. These are much stronger, and probably constitute an effective privacy-protection measure. But if you use a current-generation 802.11b network, you should treat it as under attack at all times. This means keeping your access point outside your firewall, firewalling connections from your access point going out to the Internet at large as well as into your private network (unless you want your broadband connection to be targeted by drive-by spammers), and using *ssh* or *IPsec* to create a VPN for your confidential data transfers or login sessions.

Unless your entire WLAN is buried in a concrete bunker, of course...

signal between you and any crackers can be very reassuring. Because access points cost significant amounts of money, you will want to site your access points carefully to give maximum coverage. One possibility is to place them in corridors connecting those rooms where you're going to use wireless devices – the signal is capable of carrying through a wooden door without much loss of strength, and a single access point in a stairwell is sufficient to provide coverage for most houses. Another possibility is to use a booster aerial – but be advised that in the UK there are legal limits on the strength of transmissions permitted on the 2.4GHz band, and you may be facing a hefty fine if your neighbours complain.

iwconfig usefully displays the local signal strength:

```
osh:/home/charlie # iwconfig
lo      no wireless extensions.

eth0    IEEE 802.11-DS  ESSID:"private-net"
        Nickname:"foobar"
        Mode:Managed  Frequency:2.412GHz
        Access Point: 00:30:AB:0D:XX:XX
        Bit Rate=11Mb/s  Tx-Power=15 dBm   Sensitivity:1/3
        Retry min limit:8  RTS thr:off   Fragment thr:off
        Power Management:off
        Link Quality:91/92  Signal level:-11 dBm
        Noise level:-102 dBm
        Rx invalid nwid:0  invalid crypt:0  invalid misc:0
```

See the "Link Quality" and "Signal level" headings. You can use your laptop to map a small workspace for dead spots, by simply carrying it around and typing **iwconfig** (or running *Korinoco*, which displays signal strength among other things).

If you're setting up a bigger WLAN, you may need multiple access points configured in Infrastructure mode, each acting as gateways to the same wired LAN. You can find pointers to how to configure this in the compendium of wireless HOWTOs at http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Wireless.html#howtos. For mapping signal strength on a large scale, you can set up the wireless tools and a suitable PCMCIA card on a handheld running Linux and *etherreal* (the Compaq iPaq has been used for this purpose), or use a special purpose tool such as the Locust (see <http://www.bvsystems.com/Products/WLAN/Locust/locust.htm>), designed for mapping signal strength and recording it against GPS location. **LXF**

PRESENTATION SOFTWARE

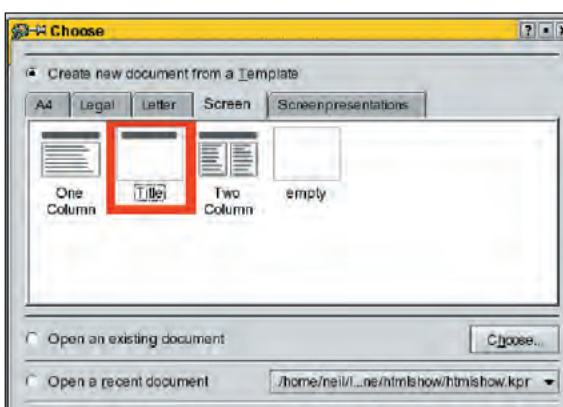
Presenting Kpresenter

KPresenter brings you the colourful world of glossy presentations. Neil Lucock shows you how to present your slides without resorting to the closed source alternative.

Kpresenter is part of the *Koffice* suite of tools. Although everyone uses word processors like *Kword*, presentation software is not something everyone has a use for. If you feel the need to stand in front of a group of people and

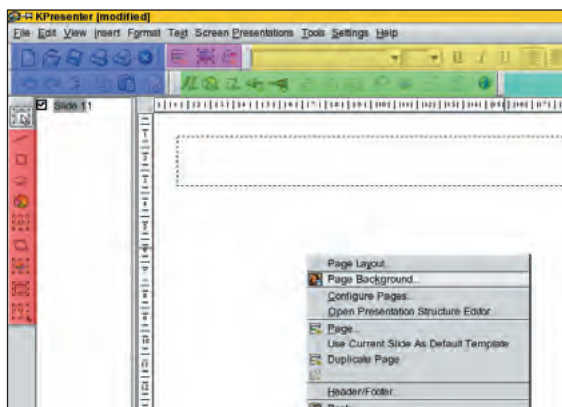
talk about something and display pictures on a screen, *Kpresenter* is the tool to use. *Kpresenter* will also run a slide show of the presentation and allow you to draw over your slides. It can make a simple HTML presentation, converting the slides to .JPG files and putting them into web pages that you can then put on your site. Like other presentation tools, it has the ability to draw simple shapes, but keeps with the "nix philosophy of "the right tool for a job". You can not draw anything complicated with it, it is better to use a dedicated drawing program and import your artwork.

1



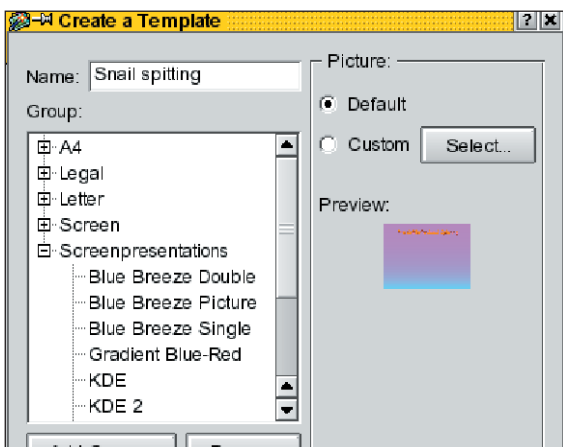
The options when you start *Kpresenter*. Launch the program and you will be presented with a dialogue box with various options to choose from. Click the tab that says "Screen" under "Create new document from template" and then select the "Title" template (highlighted in red). This launches the *Kpresenter* main screen.

2



Right-clicking on the work area for a menu. I have coloured in various toolbars. They are designed to be moved. If you want more working space on your screen, you can fold them away by clicking on the two faint grey vertical bars on the left of each toolbar. This is a common feature of *KDE* applications.

5



Save your presentation as a template. I want to re-use this background. Click the "File" menu at the top and choose "Create Template From Current Slide". I typed the name and saved it under "Screenpresentations". We want to add another slide from this template; we make it the default template in the "File" menu. To add another slide, we can use the F2 key or the "Insert" menu entry.

6



Add a logo. I want to include a picture. The block of tools that I coloured purple contains the "Insert" series of icons. The three options are: insert a new page; a picture or clipart. The clipart option imports Windows Metafiles (.wmf) clipart. The "Insert Picture" tool can put nearly any format picture into your presentation, including Windows Icons.

Color or Colour?

Beware the Yank invasion

Some might have noticed that Linux doesn't really do British English. Toolbars and menu items are spelled in American English in the program. Words like "centre" are spelled "center" in the USA, in the UK we retain the original French spelling. As this article is intended for the UK market, I've spelled according to UK English convention in the text, but quoted the names used within *KPresenter's* menus.

Half the battle with Open Source software is finding out what a program does. Many programs come without documentation, with others it is incomplete or does not describe the latest features. This tutorial will show what *Kpresenter* can do and show you how you can easily make attractive slides for your presentation. If you are trying to convince a business meeting to follow your ideas, teaching a training course or trying to explain your hobby to a village hall, it is always better to show someone something than to attempt to describe it.

I intend to discuss what the tools in *Kpresenter* can do. I assume that you have a working Linux system and that you are able to open and close files and use your printer. *Kpresenter* is found under the "Office > Presentations" entry on the KDE desktop. We will make a few slides to support our presentation and discover how each tool is used.

Kpresenter will always be compared with Microsoft's

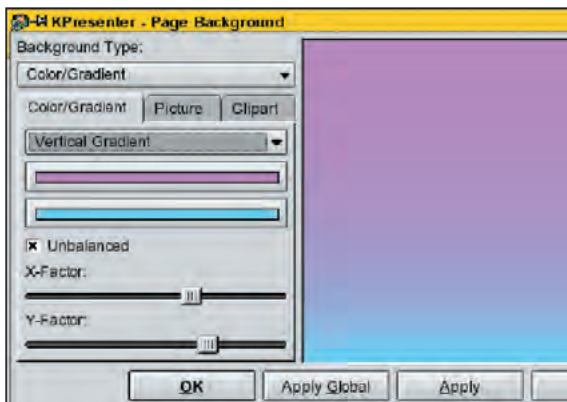
PowerPoint as it is designed to do the same job. This is not comparing like with like in a meaningful manner. One is a commercial piece of software written and marketed by the richest software company on the planet, the other is coded by enthusiasts in their spare time. The fact that you can make a comparison says a lot about the efforts of the enthusiasts. Not everything always works correctly, (I had problems with the "Group" tool), but if you keep reporting bugs, they will keep fixing them. I use *Powerpoint* at work. Could I do the same work using *Kpresenter*? Easily. *Kpresenter* does not (yet) do movies or sounds, but has all the features I use regularly. If you want to make a few slides for a presentation or training course, *Kpresenter* has everything you need.

Thanks to Southern Graphics (<http://www.southerngfx.co.uk>) for the Snail Spitting logo. No snails were harmed during the writing of this tutorial.

Import important files

KPresenter can import MS Windows Pixmaps and icons, Encapsulated Postscript Images, g3 fax files, gif, jpg, png, pnm and tiff pictures. It also supports scanning under *SANE*.

3



How to set your background colours.

Right-click on the empty part of the slide, you get a menu. Select "Page Background". Choose between a coloured background, a picture or clip art. Select "Color/Gradient" – the two bars underneath are the colours. Select colours in each "Select Color" dialogue box and click "OK". "Apply Global" will change all the slides in your presentation.

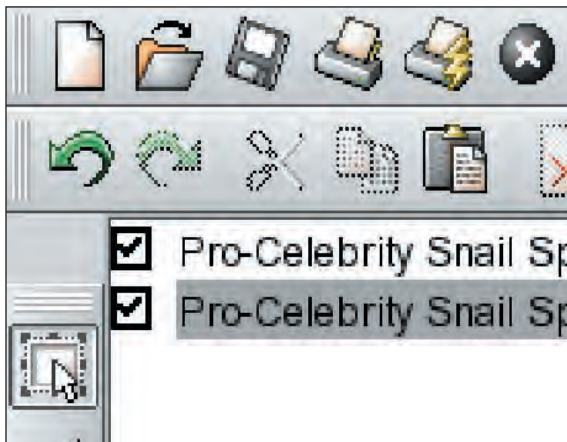
4



How to make a drop shadow.

Type in some text and click on the "Shadow Object(s)" icon (arrowed) to bring up the dialogue box. You must set the distance (between zero and eight) – you will see nothing until this is greater than one. The tool works by making a copy of the text beneath the original. A distance of two to four works well.

7



A Slide is born.

Our second slide will tell the audience what subjects we are going to cover during our talk. Make a new page (F2 key); select the default template background. Make a text insert box and type a few subject headings. We are going to make an "Unsorted list". Put the insertion point into a line of text by double clicking and use the icon on the yellow toolbar.

8



A multi-coloured unsorted list, a.k.a. bullet points.

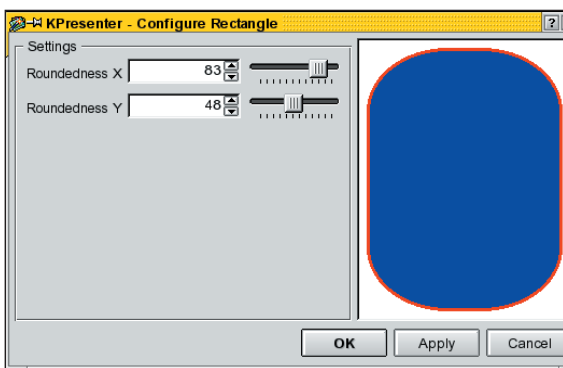
Make a new page (F2 key), a title and a text box.

Kpresenter allows you to define how your bullet points look and what colour they should be. You do this via the "Text" menu under "Settings". Now I want some ground. In the red coloured toolbar there is a "Rectangle tool". Left click and drag on the work area to make a rectangle.



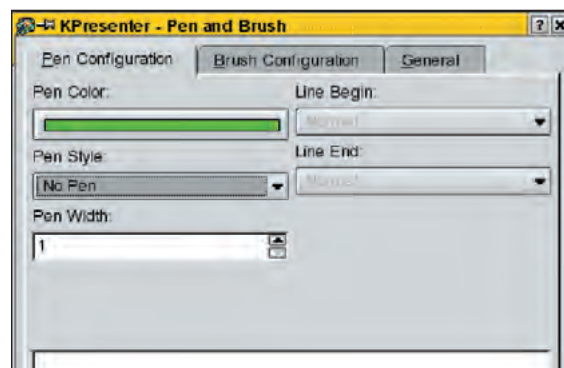
LinuxFormatTutorialKPresenter

9



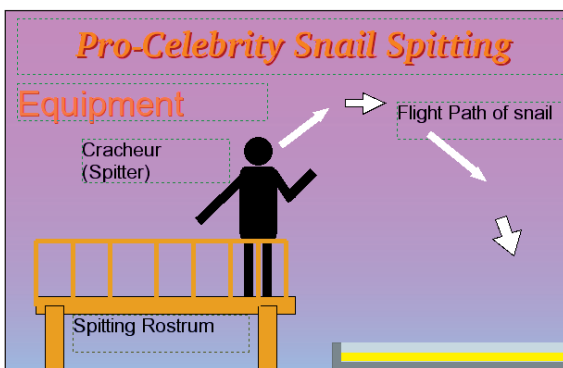
How to send a square mad. Right-click on the rectangle and a menu appears. There is a "Configure Rectangle" entry to make the shape have rounded corners. (also accessed by the square and spanner icon on the green toolbar). By moving the sliders we can make the rectangle into a circle or an ellipse. Smaller movements just round off the corners.

10



The "No Pen" setting makes the rectangle edge invisible. If you click in the black line under "Pen Color" you can select a colour for the line around the outside of the box. You can change the line width by setting "Pen Width". "Pen Style" allows your line around the box to be broken, dashed or plain. I set mine to "No Pen" as I wanted it invisible.

13



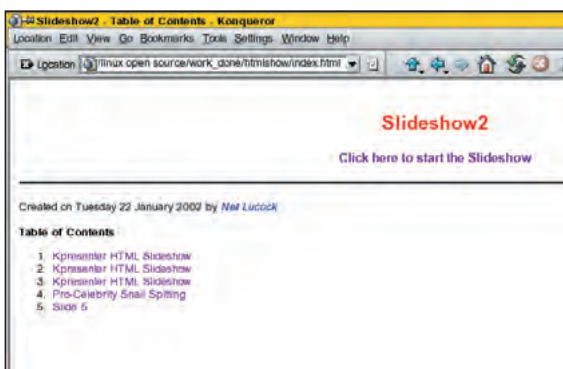
The finished Equipment slide. When you draw something new with vector drawing programs, it usually is placed in front of things drawn before it. When they overlap, the newer one is in front of the older one. If this is not what you want you can move it beneath with "Lower Objects" (on the green toolbar).

14



Boris Ulitkavich and special effects. I made four slides for my presentation. If you have a digital projector your computer can run the show for you. You use the right-arrow button on the aquamarine coloured toolbar. The slides are displayed full screen. You need to left click or touch the spacebar to advance. Right-click to get a menu.

17



The html slideshow index page in the default colours. KPresenter makes an "index.html" file, a small config file containing your details and two directories. One is "html" containing the web pages, the other "pics" containing the jpgs of your slides. The web pages are basic, but it's a nice feature. Once you have loaded it in your web browser, clicking in the web page loads the next slide in the series.

Coming next:

Improvements in the pipeline

The next release of KPresenter will have better text tools, as it will have the engine that KWord uses. It will also have a spellchecker, autocorrection and a zoom facility. On the artistic side, work is being done on including BezierCurves and freehand objects to improve the program's drawing capabilities.

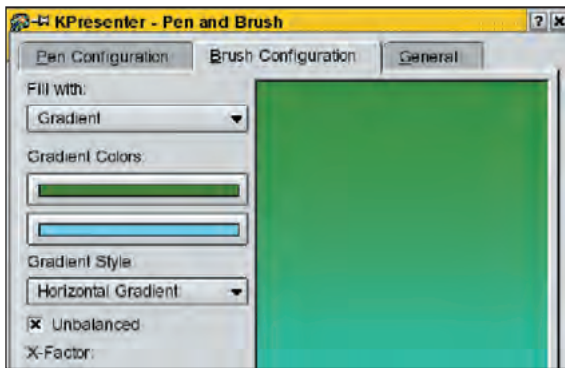
The KPresenter Toolbars

A colourful approach to presentations

The Red Toolbar: for creating things. Clicking on the mouse pointer de-selects the tool you were using and allows you to select and move things. In order, from the mouse pointer, are "create line", "create rectangle", "create circle/ellipse" & "pie/arc/chord". The one with "ab" on it is the text box tool, the slanted rectangle is the "Autoform" for inserting shapes, then "insert diagram", "insert table" and "insert object".
The Blue Toolbar: standard tools. These icons are found in nearly every KOffice program. The top row: "New Presentation", "open

an existing presentation", "Save", "print" and "print preview". The last icon is "close". The lower row of icon: "Undo", "Redo", "Cut", "Copy", "Paste" and "Delete".
The Purple Toolbar: insert page or pictures. From left to right: "Insert new page", "Insert a picture" and "insert clipart".
The Aquamarine Toolbar: on-screen presentation. Control icons - "Assign effect", "Start the show", "View the page full screen", Arrows to take you back to the beginning or end, to the previous or the next slide. The red square is the colour of the

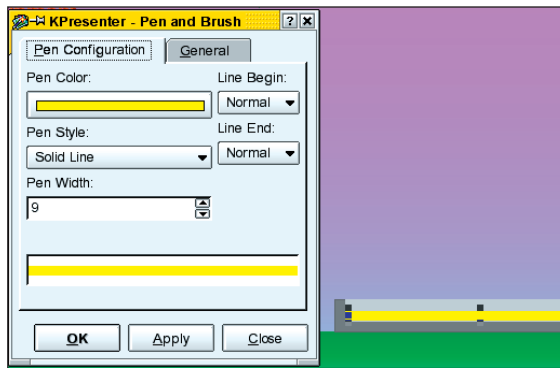
11



The gradient dialogue.

Now click on the “Brush Configuration” tab. Here we can set the colours for the inside of the box we have made. To make it look like ground I chose a “Gradient” fill. When you click on the lines to choose your colours, you can choose a recently used colour. I chose a green and the blue that I used for the bottom of the page so it blends to nothing.

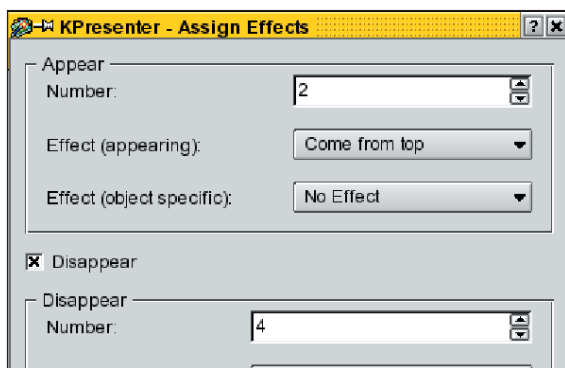
12



Colouring in a line to look exactly like garlic butter.

Now for the garlic butter that the snails land in. The second icon down on the red Toolbar is the “Line” tool. Select it, left click and drag out a line – it can always be adjusted later. Move it into position inside the garlic butter tank and right click (or use the icon on the green toolbar). Make the line thicker (about 8 or 9 is right) and colour it yellow.

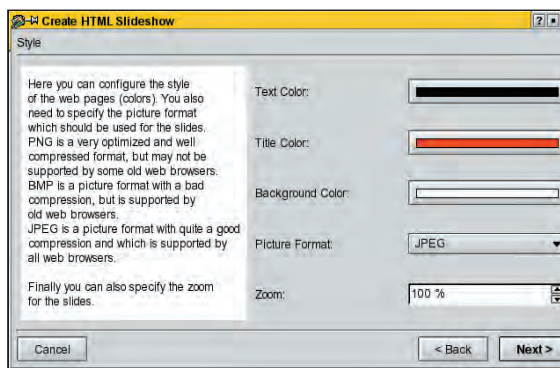
15



Make it all dance; choreography for beginners.

You can tell each object to appear or vanish at a certain time. You assign an “appear number” to it. I might want the text to be visible when the slide was displayed, then I click the mouse and Boris’s picture slides down from the top. Too many flashy transitions may distract from the presentation.

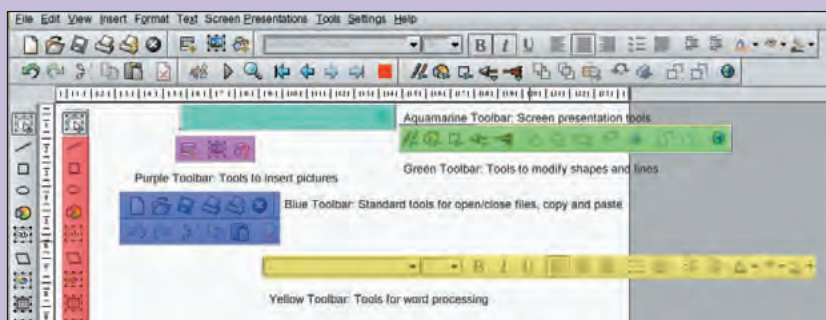
16



The default colour scheme for the html show.

On the green toolbar there is a planet icon that makes our presentation into a series of web pages. When you have entered your details, click the “Next” button to go to the dialogue where you define the style of your HTML show – jpg, bmp or png images in your web pages.

pen for drawing on your on-screen presentation.
The Green Toolbar: alterations. “Pen and Brush” to alter line and fill colours. “Configure pie/arc/chord” to alter the length of a pie segment, an arc or chord and modify them. “Configure rectangle” makes the corners rounded. “Line begin” and “Line end” make the ends of lines into arrows, squares or a star. “Raise objects” brings objects created first in front of ones made later. “Lower objects” puts new objects behind old ones. “Align objects” positions the selected object to



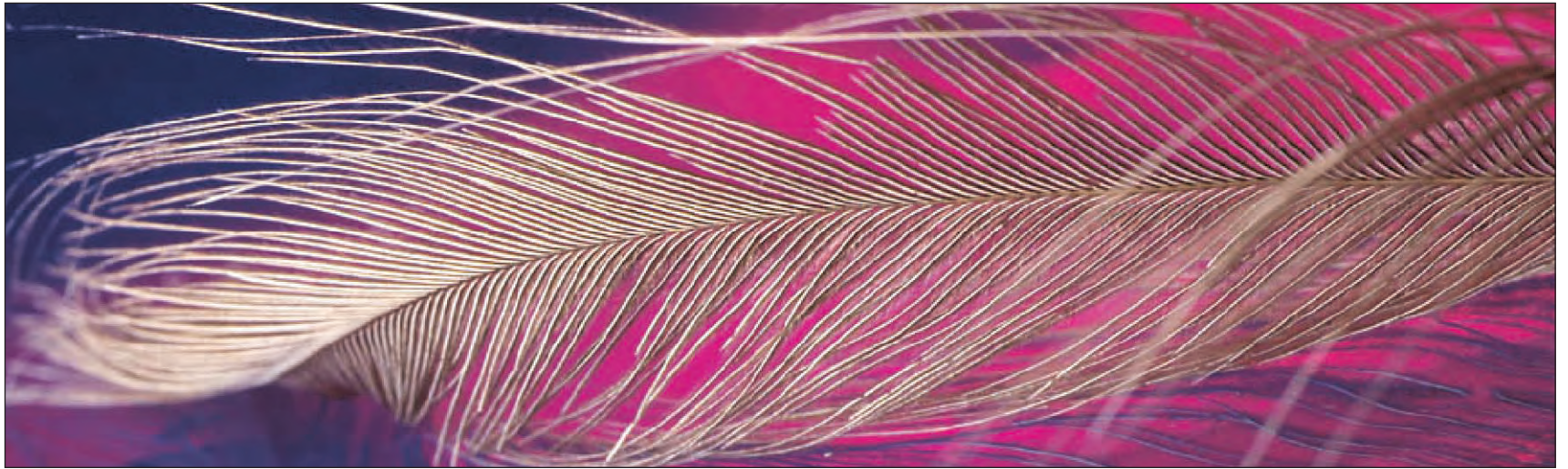
Throughout the tutorial I refer to the KPresenter toolbars by colours.

the left, right, top or bottom. There is a menu under this icon with six choices. “Rotate object” rotates them either by a fixed or

a custom amount. “Shadow object” puts a drop shadow on objects such as squares. “Group object” is used by selecting

several items using the Shift key as you click on each. You can glue them together. “Ungroup objects” splits a group apart.

“Create html show” makes your presentation into web pages.
The Yellow Toolbar: text tools. “Choose font”, “Font size”, “Bold”, “Italic”, “Underline”, “Align left”, “Align centre”, “Align right”, “Unsorted list” and “Normal text”, which gets rid of the Unsorted list. “Decrease depth” moves an unsorted list to the left, “Increase depth” moves it to the right. The underlined A icon is the text colour, the paint bucket (“Brush color”) is the flood fill to colour in a shape. The “Pen Color” tool alters line colour, including those around the outside of a shape. [LXF](http://www.linuxformat.co.uk)



THE WORLD WIDE WEBSERVER

Apache webserver

In the first of a short series of tutorials on Apache, **Chris Brown** looks at getting started with installing and configuring the ever-popular web server.



Find RPM and source for Apache on the coverdisc, as well as source for the latest Apache 2 beta.

Stable, fast, extremely flexible, and available across a wide range of operating systems, the *Apache* web server is one of the triumphs of Open Source development. Over 60% of the world's web sites, so we're told, run on *Apache*. But how difficult is it to set up and use? Well, it depends. One of the nice things about *Apache* is that it doesn't present you with a massive learning hurdle to get started. It's very easy to make it do simple things, like hosting a single, static web site. You can learn more, and add new features, gradually.

In this, the first of a four-part series, we're going to focus on how to install *Apache*, and how to get up and running with a basic, default configuration. Later articles in the series will look at the capabilities of some of *Apache*'s key modules, and how to configure them. The emphasis is on administration and config of the server; we won't be getting into how to design web content. By the end of the series, we'll know how to put together a fully-fledged web site supporting virtual hosting, password-protection, logging, server-side scripting, and a host of other features.

Choose your installation method

There are several ways to install *Apache* on your Linux system.

- 1** You can include it when you install your Linux distro.
- 2** You can install it from a binary RPM.
- 3** You can install it from source, either downloaded from an Internet archive, or from our coverdisc.

We'll look at each of these methods in turn. The table (**figure 1**) shows some of the pros and cons of each method.

Assuming you have access to a set of CDs for a standard Linux distribution, including *Apache* as part of your initial Linux installation is undoubtedly the easiest route to go. You'll not have to type any messy commands, and chances are it will simply start up in a basic configuration, and you'll be able to go right ahead and start adding web content. You'll need to spend a bit of time figuring out where and how it was installed; e.g. you'll need to know where the **ServerRoot** and **DocumentRoot** directories are (we'll discuss those later), but that shouldn't be too hard.

If, for whatever reason, you didn't include *Apache* in your initial installation (perhaps you didn't realise you needed it), your

next easiest option is probably to install it from an RPM file. You need to be logged in as root to do this. To remind you, the commands shown below are prefixed by root's special **#** prompt.

RPM (*Red Hat Package Management*.) is a system for recording what software packages are installed on a machine, tracking the dependencies between them, and handling most of the detail of installing and removing packages. Prior to installation, a "package" is stored in a .rpm file, which includes the files which make up the package, scripts to automate the installation process, and information about the package and the files it depends upon. For example, executable programs usually depend on (require) a number of shared object libraries, to which they link dynamically when they start running.

Packages are managed using the **rpm** command, which has a multitude of options for querying the packages already installed, querying uninstalled packages which exist only as .rpm files, and installing, upgrading, or removing packages (see the man page).

It didn't go quite as planned

When I started this piece of the article, I had assumed that the description of loading the *Apache* RPM would be rather brief, and the message would be "See, installing from RPMs is really easy". Well, it didn't turn out that way. So, on the basis that the only thing better than learning from your own mistakes is to learn from somebody else's, I offer the entire sad story to you.

Since I had Red Hat 7.2 installed on my machine (but no *Apache*), I mounted up CD1 of the Red Hat 7.2 distribution, and looked in the directory /mnt/cdrom/RedHat/RPMS. There I found a file called *apache-1.3.20-16.i386.rpm*. If I had not known in advance to look in the RedHat/RPMS directory, I could have done

```
# find /mnt/cdrom -name 'apache'
```

as a brute-force search, which reports two files:

```
/mnt/cdrom/RedHat/RPMS/apacheconf-0.8.1-1.noarch.rpm
```

```
/mnt/cdrom/RedHat/RPMS/apache-1.3.20-16.i386.rpm
```

The first one, *apacheconf*, is a graphical tool for configuring *Apache*, which we don't need. The second one is the *Apache* server itself. If you know how to dissect these long and scary file names, they tell you quite a lot about the package (**Fig 2**).

I decided to ask *RPM* to do a “test installation” by invoking it with the `--test` option. This pretends to do the installation and reports any errors it would encounter. I didn't expect any, but:

```
# cd /mnt/cdrom/RedHat/RPMS
# rpm -i --test apache-1.3.20-16.i386.rpm
error: failed dependencies:
libmm.so.11 is needed by apache-1.3.20-16
```

What this is telling me is that the *Apache* package needs a file called *libmm.so.11*, but that it's not installed on the machine.

OK, so where do I get this *libmm* thing from? At this point I made an incorrect assumption – that the package containing the file *libmm.so.11* would be called *libmm-something*. So I started hunting on the Red Hat CDs for files and RPM packages starting with *libmm*. To cut a long story short, there weren't any.

Puzzled, and without much hope, I installed *Apache* with the `--nodeps` option, which tells *RPM* to ignore dependency errors:

```
# rpm -i --nodeps apache-1.3.20-16.i386.rpm
```

The command completed OK, but when I tried to start the server, I wasn't terribly surprised when it complained:

```
# httpd -v
httpd: error while loading shared libraries:
libmm.so.11: cannot open shared object file:
No such file or directory
```

(The `-v` flag tells *httpd* to just report its version number). At that point I began to wonder if I could persuade the editor to take an article on something easier. Logging in, perhaps?

Rather as a last resort, I paid a visit to www.rpmfind.net/linux/RPM and did a search for *libmm.so.11*. This resulted in three references to a package called *mm-1.1.3-8.2mdk.i586.rpm*. A glimmer of hope began to, er, glim (or whatever it is that glimmers do), and I returned to my Red Hat CDs to discover that CD1 contains the package *mm-1.1.3-1.i386.rpm*. I checked that the package contained the file I was looking for with:

```
# rpm -qlp mm-1.1.3-1.i386.rpm
```

which confirmed that the file */usr/lib/libmm.so.11.0.23* (among others) was included. Once I had installed this, using:

```
# rpm -i mm-1.1.3-1.i386.rpm
```

I was then able to start *httpd* successfully:

```
# httpd -v
Server version: Apache/1.3.20 (Unix) (Red-Hat/Linux)
Server built: Sep 5 2001 23:12:29
```

The mission debriefing

It seemed a bit unsatisfactory to me that I ended up grubbing about on the *rpmfind* web site to realise that I needed the *mm* package, and I began to wonder if there was a better way to make the connection. So I mounted up my Red Hat CD1 one

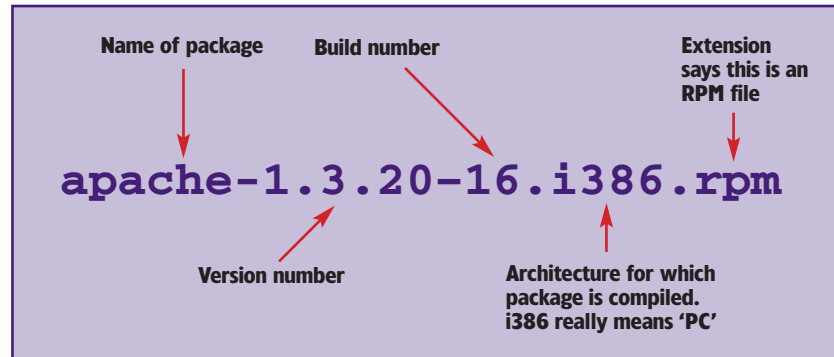


Figure 2: Anatomy of an RPM file name.

more time, changed into the directory */mnt/cdrom/RedHat/RPMS* and tried a brute force search of all the rpm files for one containing a file called *libmm-something*:

```
# rpm -qlp *.rpm | grep libmm
```

which reported:

```
/usr/lib/libmm.so.11.0.23
```

This at least told me that the required library was in one of the rpm files in the RPMS directory, but it didn't tell me which one. To find out, I created a little script called *findrpm* like this:

```
#!/bin/bash
cd /mnt/cdrom/RedHat/RPMS
for f in *.rpm
do
  if rpm -qlp $f | grep libmm
  then
    echo $f
  fi
done
```

which, when I ran it, gave the output:

```
$ ./findrpm
/usr/lib/libmm.so.11.0.23
mm-1.1.3-1.i386.rpm
```

This gave me what I needed – the link between the file I was looking for and the package it was contained in. As far as I know, there is no option to *RPM* to tell it to search all the uninstalled packages (i.e. rpm files) for the package containing a given file.

One of the things I found quite interesting about this exercise, apart from my own stupidity at confusing the names of the missing file and the missing package, was the range of tools and commands I ended up using during the course of the exercise – *find*, *RPM*, *grep*, even a bit of shell scripting. One of the things I like about Linux is having this toolbox of utilities to fall back on. Of course, you could argue that Linux would be even friendlier if the thing had installed without any problems in the first place ...



Installation method	Advantages	Disadvantages
Done as part of installation of Linux distribution	Easy, will probably work	You have to decide whether you'll need the package up front. You get whichever version is on the distro, which will probably be old. You don't have the source.
Install from a binary RPM	Quick and easy, provided there are no problems with missing dependencies. You can download and install a newer version than the one on the original distro. RPM handles dependency management for you.	You won't get the very latest version. You have limited flexibility in configuring the installation. You don't have the source. Needs a different RPM for each architecture.
Install from a source tarball	You can get the very latest version. You have total flexibility in configuring the installation. You have the source code.	Takes a little longer. Needs expert help if the build fails (but usually it's okay). One tarball handles all architectures.

Figure 1: Pros and cons of the installation methods.

«Throw away the keyboard

Speaking of friendliness, Those of you with *qwertyphobia* (fear of keyboards) might like to know that there is a graphical version of the *RPM* command called *GnoRPM*. This will provide you with a hierarchical view of the RPMs on the CD, and allow you to select the ones you want and click the INSTALL button, as shown in **fig 3**. You can also query installed packages for their descriptions and the files they contain as shown in **fig. 4**.

GnoRPM won't do anything you can't do with the command line version of *RPM*, except check that the mouse works, and it needs to be said that if you're the kind of person who shies away from the command line, you're probably going to have an unhappy relationship later on with *Apache's* terse, syntax-rich configuration file and logging output.

Now that the dust has settled

Well, having completed the installation, let's look at the files that were created. The `-ql` flag option for *RPM* asks for a list of the files in a package:

```
# rpm -ql apache-1.3.20-16 | less
```

Note that we are now querying the contents of an installed package, not an RPM file. Some files of note include `/usr/sbin/httpd`: the binary of the *Apache* server itself
`/etc/httpd/conf/httpd.conf`: the configuration file for *Apache*.

The directory `/etc/httpd` is called the **ServerRoot** directory. Note that the choice of this particular directory was made (presumably by the folks at Red Hat) when the RPM was built. It is not the same as the **ServerRoot** directory assumed by a default installation of *Apache* from source code, as we'll see later. *Apache's* configuration files, log files, and other files needed at run time, live under this directory.

You will also see a bunch of files with names like `mod_alias.so`; these are compiled modules which can be linked into *Apache* dynamically to provide extra features.

Finally, note the script `/etc/rc.d/init.d/httpd`; this is the startup script which will launch the *Apache* server (*httpd*) when the system boots.

Well, does it work?

I have never been an advocate of delayed gratification, so I'm always keen to make my newly installed software do something as soon as possible. Fortunately, there is enough of a working configuration provided in the default config file to try things out right away. Let's take a quick look at `httpd.conf`. Ours is in

`/etc/httpd/conf`, but remember that yours might be somewhere different, depending on your installation. It's a long file, but don't panic. Most of it consists of helpful comments, and right now, there are only two entries we're interested in:

```
ServerRoot "/etc/httpd"
```

```
DocumentRoot "/var/www/html"
```

(Again, you might have different directories named in your file.)

As we saw, **ServerRoot** is where the configuration and runtime support files live, and **DocumentRoot** is the top level directory where the actual web content (the HTML files) live. So for example, if I point a web browser at the URL `http://localhost/sample.html`, *Apache* will look in the directory `/var/www/html` for the file `sample.html`

If you take a look in the **ServerRoot** directory, even on a brand new installation, you'll probably find a couple of files, something like `index.html` and `poweredby.png`

`index.html` is the default file which *Apache* serves (for example if I simply enter the URL `http://localhost`) and the `png` file is a graphic of *Apache's* feather logo.

So now we're ready to start the server. We'll do it manually, just this once, like so:

```
# /usr/sbin/httpd
```

Don't panic if you get a prompt back straight away; the server automatically spawns child processes to run in the background. Finally, fire up a browser and direct it to `http://localhost`, at which point we should be rewarded with our first web page from our fledgling server.

Starting Apache at boot time

We'll be content with our default config for the moment, but before we can consider our installation complete, we need to ensure that the server starts automatically when the system boots. The RPM installation placed a startup script into `/etc/rc.d/init.d/httpd`, but we need to make a symbolic link to it from the boot script directory corresponding to the run level the machine will boot into. In my case the machine boots to run level 5 so I need to put a symbolic link into `/etc/rc.d/rc5.d`, something like this:

```
# cd /etc/rc.d/rc5.d
```

```
# ln -s ../init.d/httpd S96httpd
```

With this link in place, the *Apache* server will be started automatically each time the system boots. Note that I've assumed a distribution such as Red Hat, which uses the so-called "System V" style startup scripts. Distributions such as SuSE use a different scheme.

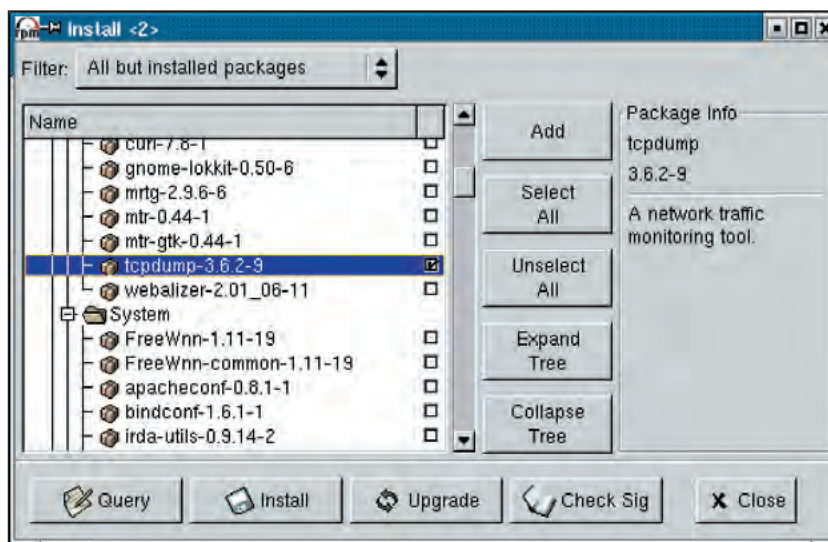
Installing from Source

If you're the type that thinks installing from source is strictly for founder members of the 'no pain no gain' school of computing, think again. Installing *Apache* from source involves compiling about half a million lines of code, but you don't need to look at a single one of them, and the entire compilation is performed by a single command, **make**. You can manage that, right? Of course, you'll need to have the C compiler (*gcc*) installed.

Installing from source offers several advantages. First, you can get the very latest version of the software. Second, you have complete control of the installation – which directories things will go in, which features will be compiled in to *Apache*, and so on. Third, there are a few important features (the secure sockets layer is a good example) which cannot be added into *Apache* without building from source. One final advantage is that once you've got the source, you can install it on any supported platform.

Notice that I'm not claiming that actually having sight of the source – in the sense of being able to take it to bed with you and

Figure 3: a user-friendly RPM installation with *GnoRPM*.



read it – is an advantage. You're unlikely to want to do that, unless you're a chronic insomniac, or a member of the *Apache* development team, or you have a very esoteric question about *Apache's* behaviour which can't be answered any other way. ("Use the source, Luke!")

Dust off that cover disk

For this exercise I decided to install a beta version of *Apache 2*, which was on the coverdisc of *LXF16*, back in July 2001. This isn't the latest version of *Apache 2*, (you can use the one on this month's disc – or download from www.apache.org) but it had the benefit of being conveniently to hand!

I mounted up the CD, and in the directory `/mnt/cdrom/developer` I found a file `httpd-2_0_16-beta.tar.gz`. (I was pretty sure that *Apache* was on the CD on account of the word '*Apache*' in big writing on the cover.) This file contains what is generally known as a 'tarball' – a compressed archive of the files making up the *Apache* source distribution. Tarballs are the standard way of packaging source distributions for Linux. Before we go any further, we need to pick a directory to install the source into. A common place to install 'local' additions is the directory `/usr/local`, so I extracted the tarball with:

```
# cd /usr/local
# tar xzf /mnt/cdrom/developer/httpd-2_0_16-beta.tar.gz
```

This command decompresses the archive and extracts the files from it. If you look in `/usr/local` you'll find a new directory called `httpd-2_0_16`, and if you look in there you'll find a file called `INSTALL` which tells you what to do next. Older versions of *Apache* used a bespoke build procedure with its own config file, but the newer versions use exactly the same build process as most other source distributions use. You simply go to the directory you just extracted the files to and run the command:

```
# ./configure
```

This is a shell script. It was built by a program called *autoconf* (One of the GNU tools) when the tarball was created. The config script probes the system to check for the presence of various features in the libraries and compiler on the system. Based on its findings, it creates some makefiles and shell scripts which will build the software on your particular platform. This very clever piece of technology makes it much easier to create source distributions that will install onto a wide range of 'nix systems.

The next step is to compile the *Apache* sources. The `configure` script has just built the makefiles and scripts needed to control this process so all you need to do is type **make**. You might be in for a bit of a wait here, because there's a lot of source code to compile. On my Pentium II machine it took about 10 minutes.

Just as a matter of interest, how much source code is there? Well, as another illustration of the power of the Linux tools used in combination, this command will tell you:

```
# cd /usr/local/httpd-2_0_16
# wc -l `find . -name '*.c' -print`
```

The answer, by the way, is a little over 500,000.

Here's another little trick. Since I was about to install the new server, I was interested to see what files were actually installed.

```
# touch /tmp/timestamp
```

created a file whose 'last modified' timestamp is the current time.

Now we do the installation – i.e. we copy all the files into the right places. You need to be logged in as root for this:

```
# make install
```

This doesn't take very long. Now we can see which files were installed, by looking for files modified (or created) more recent than our timestamp file, like this:

```
# find / -newer /tmp/timestamp
```

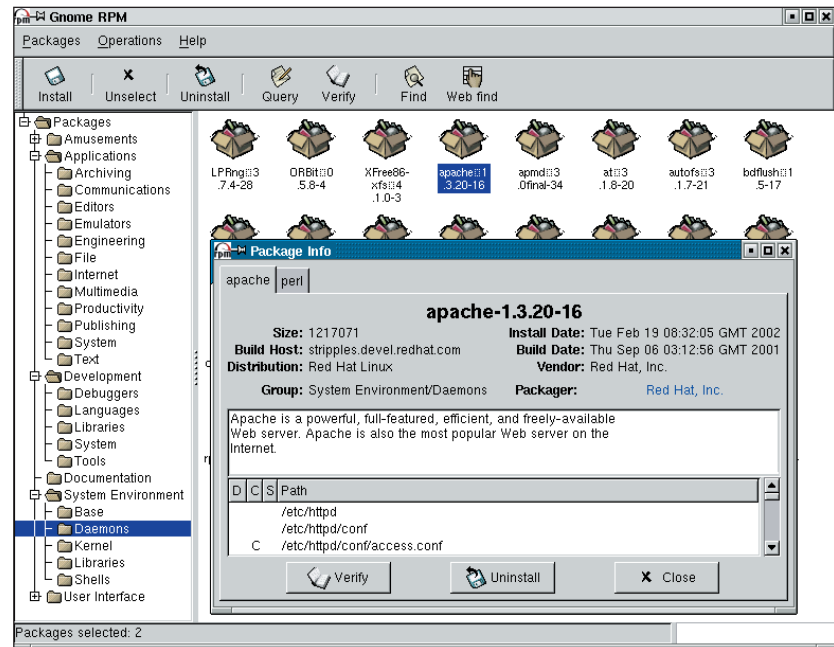


Figure 4: The querying power of the RPM command line – from a mouse click.

In addition to the output we want, this gives us a large number of spurious entries in the `/proc` pseudo file system which we would prefer to not see, so we can refine the command to filter these out, like this:

```
# find / -newer /tmp/timestamp | grep -v '^/proc'
```

You'll see some header (.h) files under `/usr/local/include`, a few libraries under `/usr/local/lib`, and a whole shedload of files under `/usr/local/apache2`, including `bin/httpd` (the server binary) and `conf/httpd.conf` (the main configuration file)

To summarise, there are five steps to installing a tarball:

1. Unpack the tarball into any convenient directory
2. cd into that directory
3. `./configure`
4. `make`
5. `make install`

There are several ways we can customise the build process.

For example, we can supply an option to `configure`, like this:

```
./configure --prefix="/usr/local/apache2_0_16"
```

in order to explicitly set the **ServerRoot** directory. I just accepted the default, which turned out to be `/usr/local/apache2`

Note that installing from source in this way does not update the *RPM* database of installed packages. For example, if I ran **rpm -qa** to query all installed packages, I would not see my newly-installed *apache-2.0.16*, nor would I be able to use *RPM* to remove it.

Getting the server started

Test your server by starting it manually, as we did after the *RPM* installation – except that now the binary is `/usr/local/apache2/bin/httpd`. Later you should put an entry into an appropriate boot-time script. In this case I chose a minimalist solution and just appended:

```
echo starting apache2
```

```
/usr/local/apache2/bin/httpd
```

to the startup script `/etc/rc.local`

This will work fine, although you won't get nice messages from the system about starting and stopping the service as you would if you integrated it in with a 'proper' startup script which adheres to the Red Hat conventions. By the way, don't try to start up more than one version of *Apache* at once. They'll fight over port 80 and the one that starts up second will lose. **LXF**

NEXT MONTH

Next month, we'll look at the architecture of the browser/server relationship in more detail, and start to get to grips with the *Apache* configuration file.

VERSION CONTROL SYSTEM

Setting up CVS

PART 2 Jono Bacon sets up a CVS server and shows you how to make good use of it.

Last issue we looked at the CVS client program to connect to a CVS server, check in some code, checkout code and browse log entries. This is all useful if someone provides a CVS server for you, otherwise you will need to set up your own – but it's not that hard, and this month we will investigate the processes involved in doing this.

Setting up a CVS server is not a difficult procedure to administer, but there is a distinct set of processes in which you can easily make mistakes that can be difficult to solve later on. We will look at these processes and how to avoid the pitfalls.

A quick recap

Before we get started, let's have a quick recap of what CVS is and what we can use it for; this will be of particular use for those people who missed last month's issue.

Imagine you are a maintainer of a project. You are in control of the entire project and how it runs. Let us now assume that after a recent announcement you made of the project, you get an email from two developers who would like to contribute some code to your project. This is great news and can increase the development pace of your application. The problem you have now is how you can manage all three developers working on the same project. One option could be for the other developers to send you patches (small files containing new code for a project), which you can then apply to your source code. This is fine until you realise that after every patch you will need to release the source code so the developers are up to date. Now, let us assume that developer 1 writes a patch and developer 2 writes a

different patch that uses some of the same source files. These two patches are fine by themselves but together there will be a conflict. The conflict would have to be resolved by you, and it may be in a part of the code you are not very familiar with. As you can see, this is a very messy situation for three developers – now imagine this for a huge project with hundreds of developers from around the world. The situation can get unbearable. This is where the wonderful world of CVS comes in.

CVS is a system that will manage the code you and your developers work on and also how you and the developers interact and change the code. CVS will not only manage the code from all the developers, but will also attempt to resolve conflicts. CVS also has the ability to revert to previously written code right back to your first commits. CVS is in general a very flexible system, and numerous projects such as *KDE*, *GNOME* and *XFree86* use it.

Installing the CVS server is a straightforward process as it comes with the CVS distribution. To install CVS on your system, please see the *Installing CVS* box.

Creating the repository

When CVS is installed on the system, we can now create the CVS repository in which all the code will reside that the developers check in and out. The first thing to decide is where you want your repository to live. To set up /cvs I first created the directory with:

```
mkdir /cvs
```

The command needs to be issued as root as it created a new root level directory. To create the repository I then issued:

```
cvs -d /cvs init
```

If the repository was created successfully, the command will take you back to prompt silently. We can now take a look at what has been created in the directory.

A quick **ls -al** shows us that we have a **CVSROOT** directory that has been created. In this directory we have the following:

checkoutlist	cvswrappers,v	modules,v	taginfo
checkoutlist,v	editinfo	notify	taginfo,v
commitinfo	editinfo,v	notify,v	val-tags
commitinfo,v	history	verifymsg	
config	loginfo	verifymsg,v	
config,v	loginfo,v	rcsinfo	
cvswrappers	modules	rcsinfo,v	

Notice how there are a number of files; some with the same name, and ,v appended. These ,v files store the version information. The normal files (without the ,v) are actual config files. We will take a look at some of these configuration files later.

It is a good idea to create a system account dedicated to CVS:

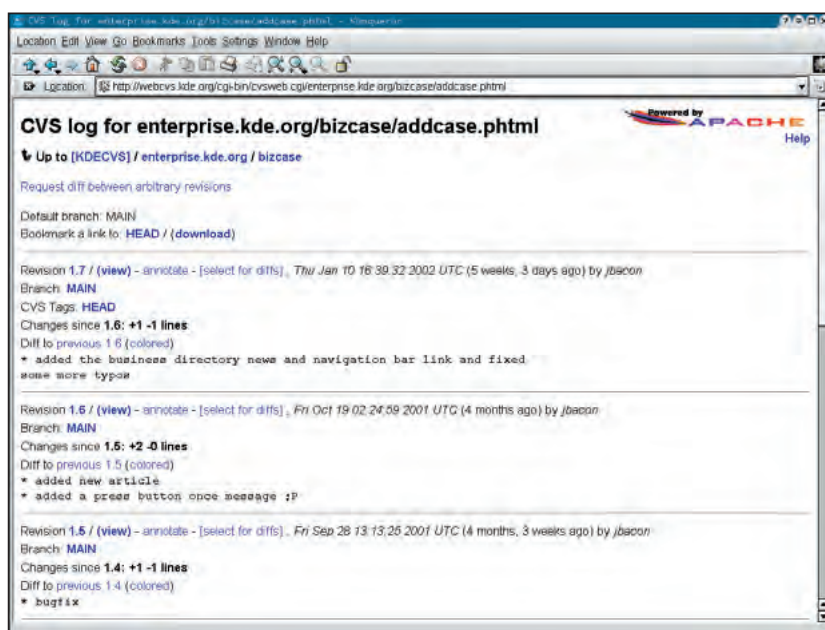
```
adduser cvs
```

Then change the ownership of the repository to this account.

Configuring the password server

It is all very well having a CVS repository, but it isn't much use without being able to connect to it securely; this is the function of the password server.

When **:pserver** is used in the **CVSROOT** of the client connecting to the machine, the client will make itself known through port 2401. The CVS server does not listen to this port actively, but *inetd* will monitor the port and start the CVS server if



CVS Web in action – it's even easier with a web-based interface.

a connection is made. To enable this to work, we first need to edit `/etc/services` and `/etc/inetd.conf`. First add the following to `/etc/services`:

```
cvspserver 2401/tcp
```

Then add the following to `/etc/inetd.conf`:

```
cvspserver stream tcp nowait root /usr/local/bin/cvs
cvs —allow-root=/cvs pserver
```

Now restart `inetd` and you should be ready.

Setting up CVS users

Although we have configured the server to accept connections from CVS clients, we need to add CVS accounts so a user can connect with a password and not compromise system security. To set this up we will create a file in `/cvs/CVSROOT` called `passwd`. The format is simple, and is in the following schema:

```
[user]:[password]:[system-account]
```

The different parts are:

user – The name of the user connecting to the server. *E.g.* `jono`.

password – An encrypted password for the user. You can copy existing encrypted passwords from `/etc/passwd` or `/etc/shadow`.

system-account – If this optional part is used, this system account name will be used for CVS operations. If `system-account` is not specified, the user account will be used. We could use the 'cvs' system account we created earlier which owns the repository.

Here is an example of mine (with the password changed):

```
anonymous::cvs
```

```
jono:fFedrR$%fDGGHtFGDF$4rDDFDF343:cvs
```

As you can see, I have two accounts – both use the system user 'cvs', and the `jono` CVS account has an encrypted password, whereas the anonymous user has no password; this would mean that the anonymous user would press enter when prompted for a password.

Anonymous CVS access is a special case as you only want the user to be able to read data from the repository and not be able to commit back to it. This can be achieved by just creating an anonymous account and restricting the permissions for that user, but this is a bit cumbersome, and CVS offers a simple system of giving read/write permissions.

To set up 'anonymous' for read only access, add 'anonymous' to the `/cvs/CVSROOT/readers` file. Each line in the file has a username for those users who can only read data from the repository. There is also a `/cvs/CVSROOT/writers` file in which people with write access can be added. All those users who are not in `/cvs/CVSROOT/writers` are assumed to have read access.

Testing the CVS

To test our setup, load up a terminal and set the **CVSROOT**:

```
export CVSROOT=:pserver:jono@127.0.0.1:/cvs
```

I covered the setting of the **CVSROOT** in the last issue, but here is a quick rundown of each part:

:pserver – This specifies that the password server is used to authenticate access.

jono@127.0.0.1 – This specifies the user and host. The user is 'jono' at the IP 127.0.0.1, which is the local loopback IP address for your machine. You can also set a domain like:

```
jono@cvs.thismachine.org.
```

/cvs – This is the system path of the repository.

Now this is set, log in with:

```
cvs login
```

and specify any password that you may have set for your user. If you are returned to the prompt, it is likely a connection was made. To test we can check out the **CVSROOT** directory from the CVS (each directory in the repository is classed as a module). Go to a directory where you will store your sources and issue:

Files from the **CVSROOT** module in the repository.

Different uses for CVS

It's not just for programming applications

CVS is often used in any place where source code needs to be managed in some way - this has in turn made CVS appear in places you would not expect to see it. This box gives some information on different uses of CVS that you may find useful:

Application Development

This is probably the most used area in which CVS is used. CVS is used to manage source code which developers commit and CVS will ignore object code, backup files and other unneeded files. CVS is at home in this situation with many projects using it for this kind of development.

Web Development

Many people don't realise how CVS can be used in web development; CVS is a handy tool which can make web development easier and more secure. A typical example is if you are writing a site in PHP and using MySQL.

All the pages can be kept in the repository and then the developer can set the Apache webroot to point to the working directory of the checked out code. This gives the developer the ability to work and test from the checked out directory and commit code straight back to the CVS server. This code on the CVS server could then be checked out to a working directory which is the webroot of the live site. This checkout can be done using *cron*, so the whole procedure is automated. I myself have developed a number of sites using this method and it works well.

Document Development

Many people who are using tools such as *DocBook* have been using CVS to manage updates to the documentation, and to create branches in documentation for different versions of the application that the documentation covers.

```
cvs co CVSROOT
```

If the files are checked out, you know the connection is working.

Administering the repository

Now that the server is set up, we can begin to configure the server and modules. To administer the server, we use CVS itself to handle the configuration files in the **CVSROOT** module. So any changes we wish to make should be made in the local copy of the **CVSROOT** and then checked back into the repository.

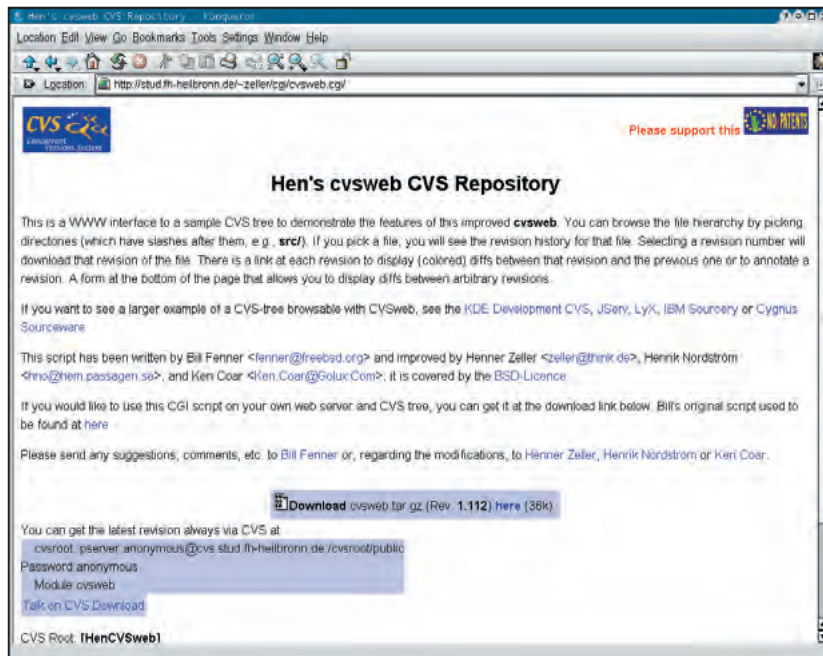
The first thing we should look at is setting up the 'modules' file. This file contains a list of modules in the CVS, and is often used by clients to see what is the repository. The format of the file is this:

```
[module-name] [path-to-module]
```

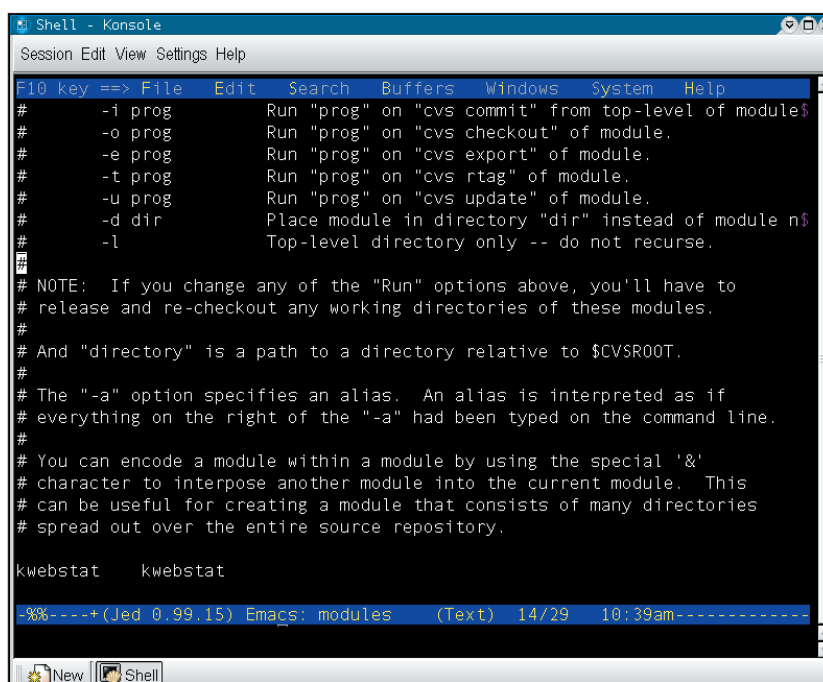
The **module-name** is the name of the particular module in the repository, and the **path-to-module** is the path within the



LinuxFormatTutorialCVS



The CVS Web homepage. Go here, read – two steps to more CVS knowledge.



The 'modules' file – often used by clients to check the contents of the repository.

CVS Web

Viewing your CVS with CVS Web

CVS Web is a CGI script which will create a web based interface to your CVS. It will allow people to view files and log entries; generate diffs and more. It is a useful tool that can help with keeping a track on what is being committed to your server.

To install CVS Web, go to <http://www.cvsweb.org/> and download the tarball. You can then install it:

- 1 Copy the `cvsweb.conf` to the Apache `$ServerRoot/conf`. Edit the file where needed,

but the most important part is the:

'Local-CVS' => '/cvs'

(if /cvs is your repository)

The other parts of the file adjust the look and feel of CVS Web as well as some other settings.

- 2 Copy the `cvsweb.cgi` to the cgi-bin of your server. Edit the `$config` variable to point to your configuration file.

- 3 Restart Apache with:
apachectl restart

repository to the module. As most people call their modules the same name as the directory, you frequently see the 'module' file entries as:

thismodule	thismodule
thatmodule	thatmodule

Once you have set the modules file up, the server is pretty much ready for use.

Adding a module

One of the first things you will want to do with your brand new CVS server is add some modules to the repository that people can start working on. This is a simple procedure and involves a single command – but before we actually import the project, we need to prepare it, and remove any unwanted files and junk that we don't want in the repository.

The only things that go in the repository should be files which are to be distributed with your project. This means that the following types of files should *all* be removed.

~file – This is often a backup left by a text editor. Check the new version of the file and make sure you are fine to delete the backup and then remove it.

Makefile.in – These are generated files from *Makefile.am* and *configure.in.in*. These generated *configure.in* files (as well as other generated files) should be removed.

file.o – This is an object code file with compiled object code in it. Remove these.

Look out for generated code from tools such as *Qt Designer* – keep the file which is used to generate the code, but not the generated code. CVS will remove some junk for you automatically.

With most projects (particularly those using *AutoConf/AutoMake*) you can remove all of this unwanted rubbish with:

make distclean

Note that you must do a **distclean** and not just a **clean** – **clean** is not thorough enough.

When you have removed all the unwanted tat from your project directory, you are ready to import it into your repository (assuming you have write permissions). Go to the directory where the project is:

cd myproject

Then issue the following command:

cvs import myproject 1_0 start

Let's just take a look at what the different parts of this command are:

import – This is the command for putting projects in the repository so CVS looks after the versioning for you.

myproject – This is where in the CVS you want it putting. By specifying **myproject** we are asking CVS to put it in the rootlevel part of the repository (/cvs/myproject). We could ask CVS to put it inside another module name (e.g for it to go in 'thatmodule' we would use `thatmodule/thismodule`).

1_0 – This is the vendor tag for that branch in the CVS. A branch is like a specifically developed version of the application. Eg. **1_0** and **1_1** are two separate branches, and hence separate changes to the same source code.

start – This is the release tag which is used to identify files for each leaf that is created by the **import** command. We can use **start** to indicate the start of development.

When you issue the command, CVS will tell you if there were any conflicts when the project was added to the repository.

Security Considerations

Security is an important consideration when working with CVS modules. You must be careful not to commit into a public

CVS passwords and other sensitive data that may compromise your system.

A typical example with this is when working on a web site. You may be using MySQL and PHP, and to connect to the MySQL database you will need to specify a username, password and host. Although this may seem a security risk to commit this information to the CVS, there is a simple workaround.

All you need to do is to create a file (which will not be in the CVS) which has some variables which contain the right data. For example:

```
<?php
    $DB_User = "myuser";
    $DB_Pass = "thepass1654";
    $DB_Host = "mysecrethost.org";
?>
```

You can then use `include()` (or whatever is similar for your language) to include this non-repository local file, and when you need to refer to the information, use the variable instead. This also gives the ability to change the information once, and it is reflected across the site (similar to a stylesheet).

Branches

Branches are a useful feature in CVS that allows you to fork development into two separate directions where each version does not affect the other. How is this useful you may ask? Let me explain with an example.

Let us assume you are working on an application called *EasyCVS*. *EasyCVS* was created initially, had the junk removed and was then imported into the repository. *EasyCVS* then undergoes some intense development as it builds up towards the 1.0 release. When the 1.0 point is reached, the code is packaged and released.

Now, as a developer you are waiting to add some snazzy new features in the 2.0 release of *EasyCVS*. You begin eagerly working on the new version and you add some major new features which require changing a lot of the code, and some elements of redesigning your classes. This is all going well until you receive an email from someone regarding a fairly serious bug in your application that really needs fixing. You have a problem here now – you have already started writing major new code which involves structure redesigns, but you know that the bugfix code will not work with the new additions. Essentially you are stuck now – do you discard the new additions in favour of the original release that is bugfixed, or do you ditch bugfixes and just go with the new features, taking time to fix the reported bug as you code.

With CVS, you don't need to make such a decision. CVS gives you the ability to branch your code into a 1_0 and 2_0 branch. When you make the branch, the code is essentially the same, but you can then make the bugfix commits to the 1_0 branch, and then add the new features to the 2_0 branch. This gives you the ability to release updates to the 1_0 release if you need to. The clever thing about these branches is that CVS records all of this information in a single module – there are no additional modules for each branch – this makes no difference to client, but to the administrator it saves having lots of extra modules.

To get started with branches, you can create a branch in the *EasyCVS* module by issuing the following in the working directory:

```
cvs tag -b 1_0
```

This creates a branch of the working directory called 1_0. You could also issue:

```
cvs tag -b 2_0
```

This would create the second branch based on the working directory also. Remember that these branches are *not* made in the working directory, they are made on the CVS server, so you

Installing CVS

First step to a CVS server

Installing CVS is a fairly straightforward process, and is similar to the installation of other packages and source for Linux.

The first thing to do is to download the relevant packages. Many Linux distributions have prebuilt packages available, which can be downloaded and installed using the package management tools in the distribution. Many distributions also provide the packages for download on their FTP sites. Check your distribution for specific details.

Another site where you can gather CVS resources and downloads is <http://www.cvshome.org/>.

When you have got the relevant packages for your system, you can install them using the following instructions:

Compiling the Source

If you got a source package, you need to first of all unzip and untar it into a directory:

```
tar xzf cvspackage.tgz
```

You can then read the INSTALL file on specific details of compiling the package, essentially:

```
./configure
```

```
make
```

```
make install
```

Installing an RPM

If you downloaded a prebuilt RPM package you can install it with:

```
rpm -i cvspackage.rpm
```

Installing from Debian

Using the Debian distribution you install with:

```
apt-get install cvs
```

```
Session Edit View Settings Help
F10 key ==> File Edit Search Buffers Windows System Help
head 1.13;
access;
symbols
    start:1.1.1.1 jono:1.1.1;
locks; strict;
comment @// @;
-
1.13
date 2002.02.06.14.18.40; author jono; state Exp;
branches;
next 1.12;
1.12
date 2002.02.06.00.39.14; author jono; state Exp;
branches;
next 1.11;
1.11
date 2002.02.05.16.45.53; author jono; state Exp;
branches;
next 1.10;
%----+(Jed 0.99.15) Emacs: kwebstat.cpp,v (Text) 1/572 10:41am----
```

The internals of a ,v file - where the version information is stored.

therefore need to check out the right branch:

```
cvs co -r 1_0 EasyCVS
```

This would check out the 1_0 branch, and all commits would be made to that branch. You are now in a position to maintain the two separate development trees.

Conclusion

CVS is a big topic; there is a lot that can be done with it, and I have tried to cover as much as I can here. There are, of course, many other facilities in CVS of which you can make use, but the topics I have covered are the major ones you should learn. I suggest visiting the CVS website at <http://www.cvshome.org/> and reading the manual and documentation if you need any further help.

CVS is something that cannot replace proper management of your team and code, but can certainly assist in it – and, with the increasing number of projects using CVS, knowledge of it is a prerequisite in many areas. Have fun! [LXF](#)

IPC AND SERVER DESIGN

Fun with servers

Charlie Stross continues his exploration of Interprocess Communications as he develops a server and client for man pages, using the `NetServer::Generic` module.

Last month we took a brisk – and highly selective – tour of Perl interprocess communications (including signals, pipes and sockets). At the end we left off with a very simple (indeed, brain dead) example of a client and a server, written using the `IO::Socket` module, which conceals most of the heavy lifting behind a halfway usable interface.

Note that if you want to understand the heavy lifting involved in socket programming, you've come to the wrong tutorial. You want to start by reading *Advanced Programming in the UNIX Environment*, by W. Richard Stevens. This tutorial is about doing it the easy way, and is not the right place to start if you want to write a high-performance web server to knock *Apache* off its throne.

Let's recap quickly. A client/server setup is one where one machine or program (the client) sends a message (possibly a chunk of data for munging, possibly a set of keystroke events, possibly a HTTP request for a web page) to another machine or program (the server). The server processes the message or data and sends something back. Why do I say "machine or program"? Because it's possible for a client program to run on the same machine as the server; for example, the *MySQL* query monitor, which lets you interactively type SQL commands to a *MySQL* database, often runs on the same machine as the database server program.

Linux provides lots of interprocess communication (IPC) mechanisms for transferring data between a client and a server program, but one main mechanism – the socket – is used for transfers across a network. (Note that both ends of such a transfer may reside on the same machine – or not. It gets confusing, doesn't it?) Sockets resemble bidirectional pipes, and Perl treats sockets roughly like a file handle that is open for reading and writing simultaneously.

Here's a really simple server that we introduced last week. It listens for connections to the local machine on port 9999 and, when one arrives, sends the current date and time to the client (by printing it on the socket connection):

```
use IO::Socket::INET;

my $port = 9999; # some unused TCP port to bind to
$server = IO::Socket::INET->new(LocalPort => $port,
                                Type => SOCK_STREAM,
                                Reuse => 1,
                                Listen => 10)
    or die "Couldn't bind to port $port: $!\n";

while ($client = $server->accept()) {
```

```
$client->print(scalar(localtime(time)), "\n");
}
```

What does all this mean?

The first point to note is that an `IO::Socket::INET` object is not a raw low-level UNIX socket; it's a Perl object that encapsulates a socket. Now would be a good time to look at the documentation for `IO::Handle` and `IO::Socket`, both of which are applicable (type `perldoc IO::Handle` and `perldoc IO::Socket`). An `IO::Handle` takes a file handle and wraps an object around it, supplying methods; `IO::Socket` provides methods for binding the filehandle to a socket and sending/receiving data, and `IO::Socket::INET` provides facilities for internet domain sockets. (Other domains exist for sockets communicating over AppleTalk, AX.25, other network protocols, or the UNIX domain – interprocess communication on the local host.)

The next thing to note is that, as with all TCP/IP communications, connections are characterised by an IP address and a port number between 1 and 65535. In this case, the `new()` method is supplied with a port to *listen* to. Any connections to the local machine on that port (in this case, 9999) will go to the `IO::Socket::INET` object we've defined. The `Listen` parameter tells `IO::Socket::INET` to set up a listen queue with ten slots for incoming connections – any more simultaneous connections will be silently lost – and to be prepared to **Reuse** the socket address. The `Type` parameter is a constant used by the kernel to identify the socket type in use – in this case a stream of packets rather than a datagram connection.

Having called `new()` (which in turn calls the low-level system call `bind()`), we call `accept()`. This tells the socket that it's to wait for connections and, when one arrives, to return a copy of the `IO::Socket` object that can be used for communication with the client that just connected. You communicate by reading or writing; in this case when we accept the client connection we simply print the current time to it.

So what's missing here? Why can't we elaborate on this and turn it into an all-singing, all-dancing server?

For starters, there's a hard limit in the `Listen` queue; if more than ten connections come in, this mini-server will simply dump them. But that's trivial compared to the more serious objection – that this solution is unidirectional. Almost every such communication is bidirectional; either the client sends some data and the server processes it and returns it, or vice versa. But both of them want to talk over the same socket. How do they know when to read and when to write?

The usual solution is to use a protocol – a well-defined set of rules governing how a client and server should communicate. The purpose of a protocol is to avoid deadlock – that is, both the client and the server waiting to read, or write a reply, at the same time (so that neither of them ever write or read the other end of the connection, leaving the session hanging). By way of example,



SMTP (simple mail transport protocol, defined by RFC821, which you can read at <http://www.ietf.org/rfc/rfc0821.txt>) is a protocol. The client connects, and initiates the session by sending a HELO message (the string HELO, followed by whitespace, the sender's domain, then a carriage return and line feed). The server is supposed to reply with a three-digit numeric code, optional whitespace, some text, and a carriage-return-line-feed line ending. (Code 220 means 'service ready'; code 421 means 'service not available'; and so on.)

After initiating a connection with HELO, the client should read the response from the server and either close the connection (if it's not active) or begin sending mail. It does so by issuing a **MAIL FROM <sender>** command, then a **RCPT TO <recipient>** command, then a **DATA** command – then printing the body of the message, indicating end of transmission by sending a line with a period on its own. At each stage, the server replies with a status code indicating whether the action was successful or not. If you're writing an SMTP client you can't take success for granted – your program should be able to abort gracefully if, for example, you send message data, write the final line, and the server replies 451 (requested action aborted) or 452 (insufficient storage space on server) or something else.

As you can see, there's more to sending a message than a simple loop like:

```
while ($client = $server->accept()) {
    $client->print("HELO FROM ", $my_hostname, "\r\n");
    $client->print("MAIL FROM ", $sender, "\r\n");
    $client->print("RCPT TO ", $recipient, "\r\n");
    $client->print("DATA\r\n");
    $client->print( join("\r\n", @message));
    $client->print("\r\n");
    $client->print("QUIT\r\n");
}
```

While this *might* work some of the time, if the **MAIL FROM** fails for some reason then the rest of the session is junk – worse, it may get into a confusing deadlock with the server, or end up doing something wholly unexpected and unwelcome (for example, if one of the lines in the message @message begins with the word "turn" while the SMTP server still thinks it's talking to a sane client).

The converse holds true at the server end of the connection. Never assume that you're talking to a sane peer or that every action they (or you) request is possible!

Another point we haven't mentioned yet is that client/server dialogues take time. If another client starts banging on the door while our server is busy digesting a large email from the first, we are going to be stuck waiting for the first session to terminate. For this reason, we normally write servers to be multi-threaded, or to fork children, to handle each dialogue with a client. When a connection is returned by **accept()** we call **fork()**; the parent goes back to waiting for another connection to come in, while the child talks to the client.

How to write a server

As a way of getting to grips with the problem, let's write a really simple TCP/IP server daemon.

The major internet protocols are a bit hairy (or, at least, too big to examine in detail in a four page tutorial). So we're going to invent our very own pocket protocol, for which we can later write a client.

By way of example, think about the humble manual page. When you type **man bash** to read the *bash* manual, the program *man* goes forth and looks for the file *bash.1*, stashed in */usr/man/man1* (or wherever */etc/man.conf* says it lives), feeds it to the *nroff* formatter and various other filters (if necessary), and pipes it to your pager so you can read it on-screen one page at a time. The standard *man* implementation on Linux has a load of other features; it can cache *nroff* output files to save work, search in multiple manual sections, and so on.

If you've got a network of machines, you may not want to install the *man* pages on all of them. Wouldn't it be useful to keep all the *man* pages on a single server, and use a tool called *man* that instead of looking for a local file could send a request to the *man* page server over the network, and display whatever it sent back?

As it happens, it isn't hard to write a server like this (although there are security issues involved in it). So we're going to write a server, and a *man*-server client.

Designing the protocol

The first step is to define the protocol our client and server talk. We're not going to clone the existing *man* program's functionality in the server – instead, the server will run *man* on a local tree of *man* pages, and accept whatever *man* delivers. So all our protocol needs to do is to deliver the equivalent of a **man** command line to the server, read the server's reply, drop the



Using NetServer::Generic

Painless management of processes, connections and timeouts

NetServer::Generic is a Perl module for writing TCP/IP servers; you can install it from CPAN (the Combined Perl Archive Network) by typing at the command line:

```
perl -MCPAN -e 'install NetServer::Generic';
```

(Note: **NetServer::Generic** relies on the module **Time::HiRes**, a high-resolution timer – this will be installed automatically if you use the CPAN command above.)

I wrote **NetServer::Generic** because I was writing client/server applications in Perl at work, and got annoyed with having to write several hundred lines of code to handle managing child processes, socket connections, and timeouts. **NetServer::Generic's** goal is to allow the programmer to focus on the important parts of a server, the subroutine that handles incoming data and does something with it.

To write a **NetServer::Generic** application, you first

decide what sort of server you want.

NetServer::Generic supports the standard UNIX forking model (where a child is forked to handle each connection). It can also run in pre-forked mode, similar to *Apache*. (It has somewhat less successful support for a non-forking mode – this works fine on lightly loaded systems but can block waiting for i/o on heavily loaded machines. And it also provides a 'client' mode, in which it issues a bunch of requests to a different server – this is supplied so that you can write tools to test another server program by bombarding it with requests.) You also need to specify which TCP port you want to listen on, whether to bind to a hostname or IP address (other than localhost), and (optionally) a list of hosts/networks/patterns to refuse access to, or a list of hosts/networks that you're willing to accept connections from. Then you throw all this data at a new

NetServer::Generic object, which does the rest. For example:

```
my $server_cb = sub {
    # subroutine that reads from STDIN
    # and writes to STDOUT
    # goes here
}

my ($foo) = new NetServer::Generic;
$foo->port(9000);
$foo->callback($server_cb);
$foo->mode("forking");
print "About to start server\n";
$foo->run();
```

Full documentation can be obtained by typing **perldoc NetServer::Generic** once you've installed it.

LinuxFormatTutorialPerl

```
#!/usr/bin/perl

use NetServer::Generic;

# minimal http server (HTTP/0.9):

sub url_to_file($) {
    # for a given URL, turn it into an absolute pathname
    my ($u) = shift; # incoming URL fragment from GET request
    my ($f) = "";    # file pathname to return
    my ($tbase) = "/home/httpd/html/";
    my ($tdefault) = "index.html";
    chop $u;
    if ($u eq "/") {
        $f = $tbase . $tdefault;
        return $f;
    }
    else {
        if ($u =~ m|^/+|) {
            $f = $tbase; chop $f;
            $f .= $u;
        }
        elsif ($u =~ m|^/+|) {
            $f = $tbase . $u;
        }
        if ($u =~ m|^/+|) {
            $f = $tdefault;
        }
        if ($f =~ m|^/+|) {
            my ($path) = split("/", $f);
            my ($buff, $acc) = "";
            shift $path;
            while ($buff = shift $path) {
                my ($tmp) = shift $path;
                if ($tmp ne ".") {
                    unshift $path, $tmp;
                    $acc .= "/" . $buff;
                }
            }
            $f = $acc;
        }
    }
    return $f;
}

my $http = sub {
    while (defined ($tmp = <STDIN>)) {
        chop $tmp;
        print STDERR "Raw http request: $tmp\n";
        my ($req, $url, $proto) = split(/\s/, $tmp);
        print STDERR "request: $req url: $url\n";
        if ($req =~ /GET/) {
            my $getFile = url_to_file($url);
            print STDERR "Sending $getFile\n";
            if (! -r $getFile) {
                print STDERR "could not read $getFile\n";
            }
            my ($in) = new IO::File();
            if ($in->open("<$getFile") && $in->autoflush(1)) {
                my $httpd_hdr = "Content-type: text/plain\n\n";
                print STDOUT $httpd_hdr;
                $NetServer::Debug && print STDERR $httpd_hdr;
                while (defined ($line = <$in>)) {
                    print STDOUT $line;
                    $NetServer::Debug && print STDERR $line;
                }
            }
            else {
                print STDERR "could not read $getFile\n";
            }
        }
    }
}
```

Gvim – vim with GUI extensions – is a great syntax-colourising editor for Perl scripts.

◀ connection, and display the reply in your pager (as defined in your shell environment variable **\$PAGER** (it's usually *less*)).

We aren't going to simply have the server execute whatever command the client sends us; that would open up a screaming security hole by allowing strangers to execute arbitrary commands on the server. Nor are we going to implement some of the *man* command's more obscure features (such as dumping a man page in typeset form to a printer). Instead, we're going to provide the following options:

-k <regexp>

Search the short descriptions and man page names for the regular expression **<regexp>**. Print out any matches.

-f <name>

Print the short descriptions of any man pages named **<name>**.

-a <name>

Print all the man pages named **<name>** (from each manual section, in turn).

<num> <name>

Print the man page **<name>** from section **<num>** – for example **man 5 hosts** prints the man page "hosts.5" from section 5 of the manual, found in */usr/man/man5*.

<name>

Print the man page named **<name>** in the first section we find.

A manpage protocol session is going to be pretty simple. The client opens a connection to the server on some standard port (I'm going to pick 8888, for no very good reason) and sends a request. The server waits for the request to terminate, then sends a reply and closes the connection. If the request isn't terminated or the connection hangs, the server should close the connection anyway after 60 seconds.

A request looks like this:

```
PAGE <manpagename>
SECTION <section_number>
END
```

Each line ends with a newline, and begins with a keyword. In this case, **PAGE** means "the following string is the name of a man page to return". **SECTION** introduces a man page section. So, to get the man page for *bash*, the client must open a connection and send:

```
PAGE bash
SECTION 1
END
```

The **<section_number>** can be a number (1-8) or the keyword **ALL**, meaning send all the man pages named **<name>** – see below for how they're delivered.

Two other requests can replace the **PAGE/SECTION** pair:

```
SEARCH <regexp>
END
```

(Which is equivalent to **man -k <regexp>**)

And:

```
DESCRIBE <manpagename>
END
```

(Which is equivalent to **man -f <manpagename>**).

The server waits until it sees an **END** line, and then parses the request. If it's confused, the server sends a status message and closes the connection. If it understands the request, the server sends a status message and then the output from the **man** command, before closing the connection. We only need two status commands right now:

```
200 - OK
500 - Bad Request
```

Each status line is terminated by a newline, then the server prints the response. A response consists of a bunch of newline-terminated lines containing the output from *man*. In the case of **SECTION ALL**, it consists of the output from *man* for each man page with the given name, glued end to end with no delimiters (just as **man -a** outputs it).

Making the server work

We cheat here and use **NetServer::Generic** (see the box, *Using NetServer::Generic*) to write our server.

NetServer::Generic is designed specifically to make this sort of simple line-based protocol easy to write. We have to do some basic setup, then write a single command, the callback executed when a connection comes in. It reads from the client on **STDIN**, and writes its output to **STDOUT**. our manserver starts out looking like this:

```
#!/usr/bin/perl

use NetServer::Generic;

$main::manprog = "/usr/bin/man";

# Simple man page server
```



```

sub make_man_request {
}

sub execute_man_request {
}

my ($cb) = sub {
    print STDOUT "500 - bad request\n";
    return;
};

my (%config) = ("port" => 8888, "callback" => $cb);
my ($foo) = new NetServer::Generic(%config);

print "$0 [$S] started on port 8888\n";
$foo->run();

```

In **NetServer::Generic**, the module does all the heavy lifting of listening for connections and managing children. When a connection from a client arrives, it spawns a child to deal with it – when it calls a subroutine called a “callback” – in this case, the closure **\$cb**. (A closure is just an anonymous subroutine with its own local variable scope – it lets us pass subroutines around as if they’re variables, and stash objects away inside them for later reuse.)

The subroutine callback **\$cb** is where the work happens; right now all it does is print “500 – bad request” on Standard Output, and exit. (In **NetServer::Generic**, anything the client sends to the server shows up on Standard Input, and to send a message back to the client you simply print to Standard Output.)

There are a couple of other skeletal items here: **make_man_request()** and **execute_man_request()**. These are utility subroutines called by **\$cb**. **make_man_request()** takes whatever parameters are passed in the request and turn them into a command line for the *man* program (named in **\$main::manprog**). **execute_man_request()** takes the command line and executes it, then returns the results to **\$cb**. **\$cb**’s job is to communicate with the client – to read request lines and accumulate a request, then execute it, and return the result. Along the way it returns “500” errors if it sees anything unexpected, or if either **make_man_request()** or **execute_man_request()** return **undef** (as they do if something goes wrong).

One point to note when looking at the source of **\$cb** is that it’s rather gnarly; there’s an eval block wrapped around most of it to trap a **SIGALRM** signal that we set to 60 seconds – this is to avoid hung connections. (See *Linux Format* 25 for a discussion of signal handling in Perl, and the use of eval blocks in this context.)

Inside the eval block, the main job of the code is to read lines from STDIN and be rude to the client if what its sending doesn’t resemble a well-formed manserver transaction. As requests come in, they’re split up into tokens and the first token is identified as a valid protocol tag. This is used as the key in a hash, and all the parameters to it are saved as an anonymous array hanging off the hash key. (In practice we only ever use element 0 of this array, but it’s there in principle if we decide to extend the protocol.)

When the main loop in **\$cmd** has read a line beginning with **END**, it calls **make_man_request()** – this checks the contents of our hash of request tags for validity and throws errors if they’re malformed. If they make sense it assembles a line to pass to the *man* command, and returns this to **\$cb**. **\$cb** then passes

this to **execute_man_request()**, which again provides an eval block wrapper around the external command – in case *man* hangs for some reason. What it returns is either **undef** (an error) or a reference to an array of lines spat out by *man*. This is then passed to the client, and **\$cb** closes the connection and implicitly exits.

There are a few other things to note about this use of **NetServer::Generic**. The first is that this server will only run on localhost at present – because we don’t tell it to bind to a specific IP address or hostname, it defaults to localhost. Next month we’ll do something about this, making it accessible to a local network.

A second issue is a bit more subtle. **NetServer::Generic** is robust and stable but not foolproof. Because it reads lines delimited by newline characters from a socket, it is vulnerable to denial of service attacks by malicious users. The timeout mechanism prevents a malicious user spawning lots of connections and filling up the server’s process table, but the use of line-oriented input means that an attacker with a broadband connection who can ram garbage down the line fast enough can make the server soak up lots of memory. One way round this is to read from the socket in raw mode; we’ll look into this later, too, along with other ways of making your servers bulletproof.

Finally, in next month’s tutorial we’ll look at how to write a client that talks to this server. In the meantime, if you run it on localhost you can type:

```
telnet localhost 8888
```

And talk to it – just type *manserver* protocol commands and it will answer!

```
PAGE bash
```

```
SECTION ALL
```

```
END
```

or

```
DESCRIBE bash
```

```
END
```

Have fun! 

Controlling servers on Linux

How to keep track of your child (processes)

On UNIX systems, most network services are provided by programs that listen on a port until a connection comes in, then spawn a child process to talk to that specific connection (see article for more details). But what controls the startup and shutdown of such servers?

There are generally two mechanisms for running servers under Linux; from the *rc* scripts (executed when the system changes run level, at startup or shutdown), and from *inetd*, the internet daemon.

Servers launched from the *rc* scripts typically have a shell script (in */etc/rc.d/init.d* under Redhat, or */etc/rc.d* under SuSE or Debian) which expects a parameter, ‘start’ or ‘stop’. Executed with ‘start’ as an option, the server starts up, binds to the port directly, and listens; when ‘stop’ is given, the shell script sends the server a message to shut down. These are known as ‘standalone’ servers and they do all the child management themselves – servers written with **NetServer::Generic** fall into this category.

Services which aren’t used very often can use *inetd* instead. *inetd* is a special server that listens to a whole bucket-load of TCP/IP ports simultaneously. When it sees a connection on port <X>, it looks in the file */etc/services* to see the name of the network service associated with this port. It then scans the file */etc/inetd.conf* for an entry for this service, and forks a child that executes the command found in */etc/inetd.conf*. This is typically a simple server that reads from standard input and writes to standard output – like the callback in a **NetServer::Generic** module.

The disadvantage of using *inetd* to launch services is that it spawns lots of processes for every connection – a child of *inetd*, and then a *tcpd* wrapper, and then a server process to handle the connection – which adds to the system load. Standalone servers are inherently more efficient, which is why toolkits like **NetServer::Generic** exist to make writing standalone servers easier by providing all the networking and child management code.

OBJECT SERIALIZATION

Speaking Java

Find out how to give your objects an independent existence outside of your Java programs in this guide by **Richard Drummond**.



The source code to accompany this article can be found in the Magazine section of the coverdisc.

The Java serialization API allows the state of an object to be written to a Java stream and an exact copy of that object recreated at a later time, or even on a different Java virtual machine. This mechanism works with any kind of byte-based stream in Java. For example, you can serialize to a file-based stream to implement object persistence and allow the state of your objects to persist between different executions of a program. Or you can serialize to network sockets and so send objects over the wire to a remote Java process – this is the basis for Java RMI (Remote Method Invocation), Java's CORBA-equivalent for distributed programming. Serialization is also heavily made use of in Java's component architecture, JavaBeans, as we will see in a later issue.

Serialization basics

Any class that implements the `java.io.Serializable` interface is serializable by Java. The **Serializable interface** is what is known as a marker interface: it declares no methods and is simply used to indicate some behaviour of a class. In this case, it simply tells the serialization mechanism that the class which implements it can be serialized.

Most of the standard class library is serializable, including the GUI components. In some cases, though, it just wouldn't make much sense to serialize objects, such as, for example, objects of **File** class. The file represented by your **File** object could be modified, moved or deleted between serialization and deserialization of the object, thus rendering any information stored in the serialized object useless. Then, some objects would be just too difficult to serialize, for example, **Thread** objects.

Making a class serializable generally takes very little work on the part of the programmer. The default serialization mechanism knows how to interrogate a class and so write any of its serializable fields to a stream. The default behaviour can be overridden – and you will want to do this in some cases – but we will discuss that in more detail next issue.

When an object is serialized, a complete graph of objects is dumped to the stream; that is, its state and the states of any objects it refers to are written out. (If an object is referred to more than once, cycles are avoided by only storing that object once.) When an object is deserialized, all the objects it refers to are recreated also.

As you might imagine, this is a particularly powerful mechanism. It allows us, for example, to create a deep copy of an object by serializing and deserializing to a pipe or memory stream. The `Object.clone()` method – by comparison – only duplicates the object itself, not any object referred to by member fields. Using serialization to perform deep copies is illustrated by the example class on the coverdisc.

The default serialization mechanism is implemented by the classes **ObjectOutputStream** and **ObjectInputStream** in the package `java.io`. These are wrappers for the basic stream classes, and are used just like any other stream class. The former provides the method `writeObject()` to write a graph of objects to a stream and the latter `readObject()` to read those objects back in and instantiate them. These classes also provide methods to read and write primitive types.

Suppose we have a class **myClass** that implements **Serializable**. Serializing an object **my_object** of that class to a file, for example, is laughably easy:

```
ObjectOutputStream out = new ObjectOutputStream( new
    FileOutputStream( "tmp/obj.ser" );
out.writeObject( my_object );
```

Deserializing an object is equally simple:

```
ObjectInputStream in = new ObjectInputStream( new
    FileInputStream( "tmp/obj.ser" );
MyClass new_object = (MyClass) in.readObject( );
```

Note that when we deserialize here a new object of type **MyClass** is created. However, `readObject()` returns a reference simply to an **Object**, so this reference must be casted.

How does Java know what the class of an object is when you deserialize it? Well this information is stored in the stream along with the state – and an identifying version number. This version number is a 64-bit hash value automatically generated from the the name of the class, its fields and methods. Classes can only deserialize objects which share its serial ID. Hence changing a class – even subtly – can cause the serial version number to change and so create incompatibilities (see box *Evolving serializable classes*).

An example

Enough of the theory! It's time for an example. Have a look at the class **SerDemoGUI** on the coverdisc. This implements a simple AWT GUI with some widgets and provides a series of buttons which, when clicked, cause the window to save its state to a file, to load its state from a file, and to create a copy of itself and its current state. Copy this class from the CD to your hard drive, compile it, run it and have a play around with it.

Since this is only example, it is written in a rather naive way, and AWT components are not particularly suited to serialization – but it serves our purpose to illustrate a few points. First note that this class relies on the standard serialization behaviour, that is, it simply writes itself with `writeObject()` and deserializes itself with `readObject()`. This has a few interesting side effects

Externalizable classes

When you want complete control of the serialized data

The **Serializable** interface has a partner called **Externalizable**. Classes marked as **Externalizable** are still serialized, but only the ID of the class is written to the stream by the serialization algorithm. It is the responsibility of the class itself to store the actual state of its instances. This is done by implementing the methods `writeExternal()` and `readExternal()`. You would mark classes as **Externalizable** when you want complete control of how the state of an object is written and the format of the data stored in the stream.

Evolving serializable classes

How to maintain backwards compatibility

As we mentioned in the main article, the serial version ID of a class is generated automatically and depends on the name, fields and methods of the class. If we need to change a class and maintain compatibility with serialized data from older versions of the class, we need to explicitly declare this version ID in the class. First, you need find out the version number of the old class. Do this with the **serialver** tool provided with the JDK.

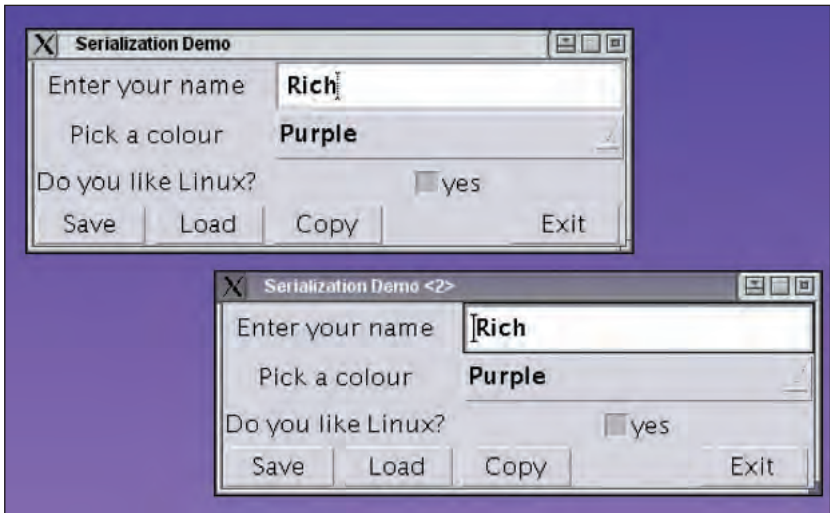
```
serialver SerDemoGUI
```

This generates an output of the form:

```
static final long serialVersionUID = 5357049008671004685L;
```

Now include this line in your updated class declaration.

Of course, this will only work if the data stored in the serialization stream is not changed. Adding or removing serializable fields to a class will break compatibility.



as we shall see in a minute.

The following code extract is executed by **SerDemoGUI** when you hit the 'Save' button:

```
ObjectOutputStream out = new ObjectOutputStream( new
    FileOutputStream( "GUI.ser" ) );
out.writeObject( this );
out.close();
```

This quite straightforward and simply serializes the **SerDemoGUI** object – the frame and all the serializable components contained in it – to a file called 'GUI.ser'.

When you hit the 'Load' button it will execute the following code:

```
ObjectInputStream in = new ObjectInputStream( new
    FileInputStream( "GUI.ser" ) );
SerDemoGUI gui = (SerDemoGUI) in.readObject();
this.hide();
gui.pack(); gui.show();
in.close();
```

What's happening here? Well, we open the file "GUI.ser", wrap it in an **ObjectInputStream()** and try to generate an object graph from it with **readObject()**, thus recreating the frame and all of its components. The reference to the object instantiated from the stream is cast to a **SerDemoGUI** object, so that we can usefully play with it. Then we have to perform some trickery. At this point we now have two **SerDemoGUI** objects instantiated: the one referred to by **this** and the one we just grabbed from the file. To perform the illusion of loading the saved state into the existing window, we hide the current window and show the new one. Of course, both windows still exist in memory, and you wouldn't do things like this in the real world.

Note that the newly deserialized window isn't visible on screen when first loaded – we have to explicitly invoke **show()** to open it. We also call **pack()** on the new frame to make it set its size to its preferred dimensions.

Now, the dimensions of the frame are actually stored in the serialized file, but these are not generally very useful to us since they are dependent on platform-sensitive aspects like the width of the frame's border, the default font size, and so on. On X, the behaviour is also very dependent on the window manager being used. Even serializing and deserializing a frame on the same window manager can have unexpected results. We get around these problems simply by resetting the window size with **pack()**.

The window's position is also set from the serialized file, and the window will appear at the same place on screen as it was when it was saved or copied. Again window position is platform-

dependent, and we would want to alter this behaviour.


If you have tried out this example, you have probably spotted another problem. When you instantiate a new **SerDemoGUI** object – either by loading from a file or copying – then the close widget on the window frame no longer works. Why is this? Well, the close window handler is implemented as an anonymous inner class of type **WindowAdapter** that is instantiated and attached to the window in the class's constructor. The **WindowAdapter** class itself is not serializable, so is not re-created in the deserialized object, and, since the constructor isn't called when you deserialize, the newly-created **SerDemoGUI** window has no window close handler.

Our example class this month demonstrates Java serialization.

Custom serialization

As you can probably tell, the serialization behaviour of the **SerDemoGUI** class is far from ideal. Part of the problem is just poor design, but part of the problem is that we have no control over how the various superclasses serialize their state. We really need to take control and customize the way that the class is serialized: we'll take a closer look at this sort of thing next month (actually, a better idea with our example class would be just to go back to the drawing board!).

One method by which you may tweak how a class is serialized is by flagging the fields that you don't wish to be serialized with the modifier **transient**: the serialization algorithm dumps all of class's fields which are serializable apart from those declared to be **transient**. This is incredibly useful. For example, if a field is only temporary, or it contains sensitive information that shouldn't be written to a file, or it contains platform-dependent data, then we can use this mechanism to stop it being serialized. It doesn't help our example much, because the troublesome fields are members of superclasses such as **Component**.

Until next time, you can find out more information on serialization by reading the Serialization specification at <http://ftp.java.sun.com/docs/j2se1.4/serial-spec.pdf>. 

Serializing to XML

How to store objects in a portable way

The default serialization mechanism writes out a binary representation of objects which is unintelligible to anything except to Java. If you wish to serialize in a portable, human-readable way, why not have a look at the Java

Serialization to XML (JSX) project (see <http://www.csse.monash.edu.au/~bren/JSX/tech.html>). This is just as it sounds: it builds on the standard serialization API to let you read and write objects as XML data.



DEALING WITH ERRORS

Exception handling

PART 8 This month **Brian Long** looks at how to add error handling to your Kylix applications

A natural part of writing any useful application is writing the error handling code. Different programming languages offer different capabilities in this regard and both this and next month are going to be spent looking at what Object Pascal in *Kylix* (and also in *Delphi*) brings to the table.

Error Handling Without Exceptions

In many older languages, such as C, the normal job of detecting errors and responding to them is entirely down to the user. As you write your lines of code many of them will involve function calls. Most functions will validate their input and behaviour and will return special indicative values if something goes wrong (such as -1 or False). Sometimes they will also set a global error variable to a value that indicates specifically what went wrong.

This is just what the Linux programming API (as defined in the *glibc* library and made available to *Kylix* programmers through the *Libc* import unit) does. Failing Linux system routines return -1 and you find the actual error code from **errno**.

As a consequence, any line of code that calls such a function is required to be written in conjunction with a conditional statement to cater for both normal operation and the error situation. As you write more code you must also write more and more conditions and the logic you are trying to implement can become somewhat obscured thanks to all this branching code. This in itself can lead to maintenance issues as the code and error handling gets more involved.

Something that might make all this more manageable is if the onus for executing the error handling code was taken away from the programmer, and was left with the language itself. If the language could recognise when an error occurred and jump straight to the error handler the programmer's logic would be more straightforward and clear cut (and more maintainable). The whole mess of conditional statements that ensure the right code is executed at the right point, depending on whether an error occurs or not, would disappear. The programmer would be left

with the job of writing simple statements, one after the other, without having to constantly check to see if an error occurred in the previous statement or not.

This is where *exceptions* come into the story.

A Recap On Kylix Error Handling

Before looking at how exceptions affect error handling, let's have a quick look at an application that has some errors in it. On this month's disk you'll find the *AutomaticErrorHandling.dpr* project that looks like **figure 1** when running.

Each of the five buttons is deliberately set up to cause a completely different type of error, and you can see the button event handlers below.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Tag := StrToInt(Application.ExeName)
end;
```

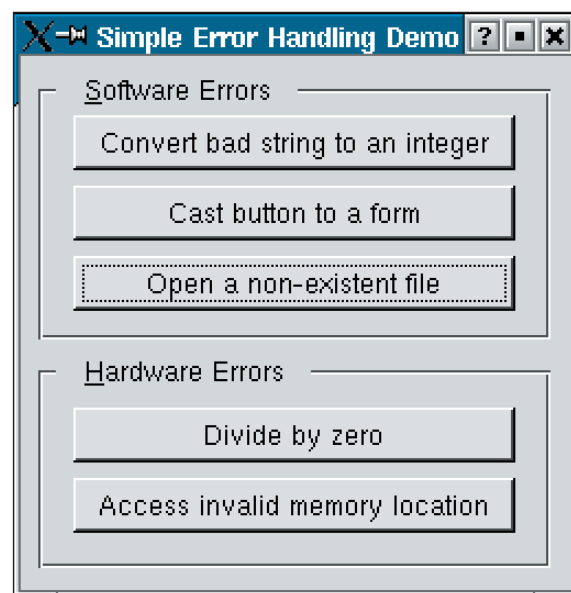


Figure 1: A simple app to show automatic error handling.

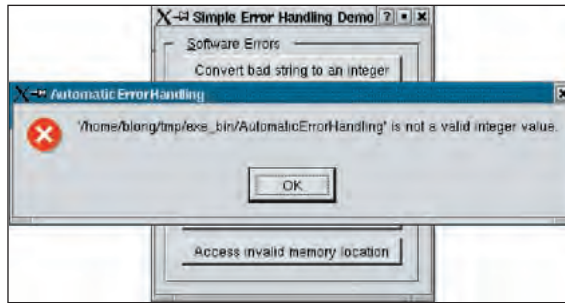


Figure 2: Your program dealing with an error all by itself.

```

procedure TForm1.Button2Click(Sender: TObject);
begin
  (Sender as TForm).Hide
end;

procedure TForm1.Button3Click(Sender: TObject);
var
  Lst: TListBox;
begin
  Lst := TListBox.Create(Self);
  Lst.Parent := Self;
  Lst.SendToBack;
  Lst.Items.LoadFromFile('Non-existentFile.txt');
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  //SIGFPE signal
  Tag := 100 div Tag
end;

procedure TForm1.Button5Click(Sender: TObject);
var
  PI: PInteger;
begin
  PI := nil;
  //SIGSEGV signal
  Caption := IntToStr(PI^);
end;

```

The first button uses **StrToInt** to turn a string representation of an integer number into an actual integer value. The result is to be assigned to the form's **Tag** property, but this will never actually happen as the source string, **Application.ExeName**, will never represent a valid number (it holds a fully qualified path to the application's executable file. Incidentally, all components have an integer **Tag** property, which defaults to **0** and can be used for any purpose you like (the **CLX** library never uses **Tag**).

The second button takes its event handler's **Sender** parameter (a reference to the object that triggered the event, which will actually be a **TButton**) and erroneously tries to dynamically cast it into a **TForm** reference in order to call the **Hide** method.

The third button dynamically creates a list box component which appears on the form. To ensure it doesn't obscure the buttons it is sent to the back of the Z order (it is drawn first and then the buttons will be drawn over it). Then it tries to fill the list box with lines from a non-existent text file, which causes the error.

If you test these three buttons, each one causes an error and the program reports it in a simple message box (shown in **figure 2**). When you clear the message box the program

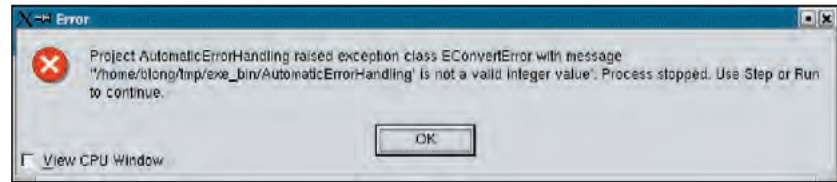


Figure 3: The debugger telling you about an exception in your program.

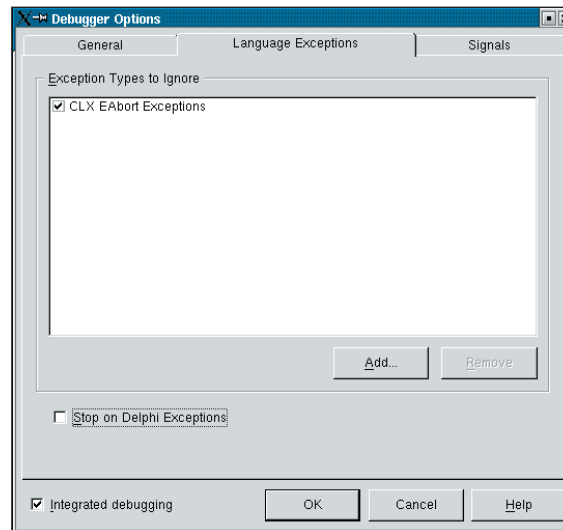


Figure 4: Stopping the debugger from intercepting software exceptions.

carries on unharmed.

Note that if the default debugging options are still set, the debugger will intercept each error and inform you of its presence (as you can see in **figure 3**), suspending the program as it does so (you can choose **Run | Run** or press **F9** to let the program continue after closing the debugger's message dialog). You can stop this from happening if need be by choosing **Tools | Debugger Options...**, selecting the **Language Exceptions** page and unchecking the **Stop on Delphi Exceptions** checkbox (see **figure 4**).

The point of this application is to show that when any of these completely different types of errors occur your application will report the error in a consistent and recoverable manner. This is without you having to write any code or even realise the errors might occur.

This first group of buttons all generate software errors (logical errors in the program) but the next two take it a stage further.

Hardware Errors And Signals

The fourth button performs an invalid arithmetic operation (a divide by zero). To get this error to happen it is not possible to simply write an expression such as:

```
100 / 0
```

or

```
I div 0
```

as the compiler will instantly reject such statements (it checks for division by a literal value of **0**). So instead the **Tag** property is used again (which, if you recall, defaults to **0**).

The difference between the **/** and **div** operator, which both do division, is the type of value generated. **/** generates a floating point result, but you cannot directly assign a floating point value to an integer variable or property (the compiler refuses due to the fact you would lose the fraction part of the result). If you really wanted to, you could pass the result to either the **Trunc**



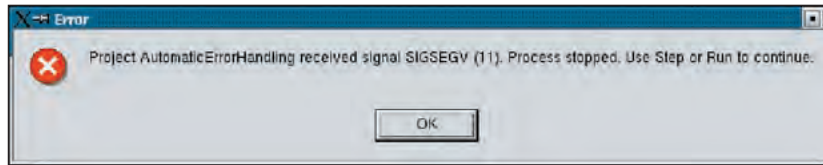


Figure 5: The debugger informing you of a signal received by your program.

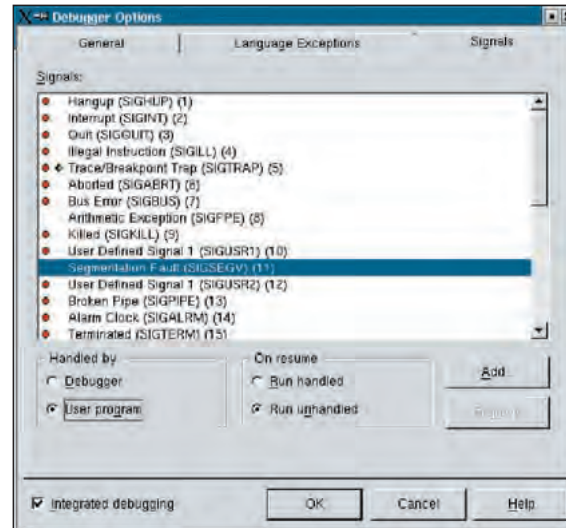


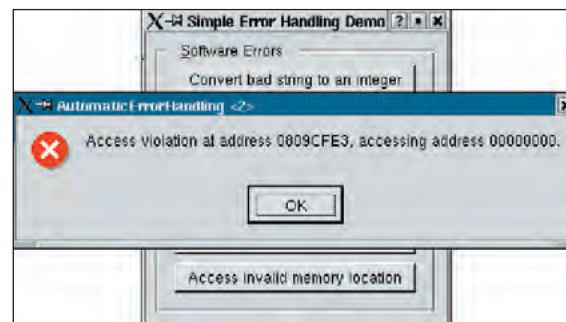
Figure 6: Stopping the debugger's interest in signals.

◀ or **Round BaseCLX** routines to truncate or round the value to an integer. The general approach, though, is to use the **div** operator instead, which performs integer division. The complimentary **mod** operator can be used to get the remainder, so **5 div 3** gives a result of **1** and **5 mod 3** gives a result of **2**.

The divide by zero error is actually picked up by the processor and so represents a hardware error. Linux reports the error to the application as a **SIGFPE** error.

The last button is designed to induce a nastier type of error. It declares a local variable that is defined as a pointer to an integer (if you are unfamiliar with the use of pointers don't worry too much about it as most *Kyl* applications don't have cause to use them). The pointer is set to **nil** (meaning it holds address 0) and then de-referenced to get an integer to pass to **IntToStr**.

This means the integer that is supposedly in memory at address 0 is read and passed to the function. However, when the CPU is running in protected mode (as it is in Linux and Windows) it performs extra validation of certain machine operations. For example, every memory access is examined to ensure the program has appropriate rights to do it. If this

Figure 7: A *Kyl* program reporting an Access Violation.

is not the case, the CPU raises a special hardware signal to stop the program from proceeding any further, which Linux translates into a **SIGSEGV** signal.

No applications have permission to read from memory address zero so this application will fall foul of the CPU's watchful eye.

When an application running under the debugger receives a Linux signal, the IDE again pops up to alert the developer (see **figure 5**). You can kill the debugger's interest in signals in a similar way to how we did it for language exceptions. In the debugger options dialog, the Signals page lists all the known Linux signals. You can select those of interest here (**SIGFPE** and **SIGSEGV**) and change their Handled by option from Debugger to User program (see **figure 6**).

Unfortunately, *Kyl* 1 seems to have a problem with its signal handling in that signals tend to kill applications dead (left hanging) and don't get very far in the debugger either (the signal report will keep recursively appearing if enabled, otherwise the debuggee will hang).

Fortunately, *Kyl* 2 seems to have better signal handling. The debugger will still notify you when a signal occurs (if the option is enabled) but the application then continues to handle the error quite successfully, reporting it as an **Access Violation** (see **figure 7**). *Kyl* 2 Open Edition is available for download from Borland's Web site (<http://www.borland.com/kyl/tryitnow.html>) and should also be available on the *Linux Format* coverdisc this month.

Error Handling With Exceptions

Object Pascal (and other languages such as C++ and Ada) support the notion of exceptions. Exceptions are representations of errors that are understood by the language itself. In Object Pascal all exceptions are objects that sit in a branch of the class hierarchy. The root exception class is **Exception** and many other exception classes inherit from it such as **EConvertError** and **EInvalidCast**. The base exception class happens to start with an **E** and all the inherited classes use an **E** prefix by convention.

When code is executing and an exception occurs to indicate a problem exists (we'll see how to do this in your own code next month) the execution automatically jumps to the closest error handler for that exception. As mentioned before, this makes the job of writing code much more straightforward and lets you forget about the constant checking for errors that other languages oblige you to make.

As you have seen, if you don't write an error handler the program responds with a default message box. If you want to respond to the error in a different way the language has constructs to achieve this.

The inclusion of exceptions in Object Pascal makes writing a robust application much simpler than it could be. When you want to write error handling code, it is separated from the logic to which it applies – making both the logic and the error handling easier to read and maintain.

The Syntax Of Exception Handling

You handle specific exceptions in a code block using a special compound statement that comprises two sections. The first section in the **try..except** statement contains the code you wish to handle errors for and the second holds the error handlers themselves (called *exception handlers*). Exception handlers each respond to exceptions of a specified class type (and those inherited from that class) so you need to know the exception to handle in advance (when the debugger

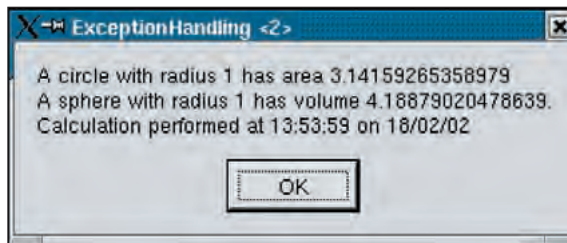


Figure 8: Correct behaviour from the app.

responds to an exception it tells you this information as you can see in **figure 3**).

What you do in the exception handler is entirely up to you, but it could be as simple as displaying a different message box with a more user-friendly error message. For example, consider some code that gets a radius value from the user and then displays the area of a circle with that radius and also the volume of a sphere with the same radius. You could write it like this (the code is in the *ExceptionHandling.dpr* project on the disk):

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Radius, Area, Volume: Double;
  RadiusStr: String;
begin
  RadiusStr := '1.0';
  if InputQuery('Trigonometry 101', 'Enter a radius', RadiusStr)
  then
  begin
    Radius := StrToFloat(RadiusStr);
    Area := Pi * Sqr(Radius);
    Volume := Pi * Power(Radius, 3) * 4/3;
    ShowMessageFmt(
      'A circle with radius %g has area %g'#13#10 +
      'A sphere with radius %g has volume %g'#13#10 +
      'Calculation performed at %s on %s',
      [Radius, Area, Radius, Volume,
      TimeToStr(Time), DateToStr(Date)])
  end
end;
```

InputQuery displays a simple entry dialog with a caption and prompt as specified by the string constants passed to it and an edit control with the value of the third string parameter written in it. If the user presses OK their new value is returned in this variable and **InputQuery** returns **True**, otherwise it returns **False**.

ShowMessageFmt takes the same parameters as **Format**, which builds a formatted string, and passes it to **ShowMessage** to display it, as shown in **figure 8** (in fact there is also an overloaded version of **ShowMessage** that takes the same parameters). **Format** was used in the code last month, and is a convenient way to build formatted strings. You supply a template string with placeholders to represent different types of values (such as **%g** for a floating point number and **%s** for a string) and

then supply a list of expressions whose values get plugged in. You can find more information about these placeholders in the help (look up *Format function* and *Format strings*).

The **Power** function and many other useful maths routines are implemented in the **Math** unit so you must add **Math** to your uses clause before this code will compile.

StrToFloat generates an **EConvertError** exception if it goes wrong, such as being passed a string that doesn't represent a float) and the default error message look like **figure 9**. You could choose to make a more friendly error message by changing the code to this:

```
uses
  Math;
...
procedure TForm1.Button1Click(Sender: TObject);
var
  Radius, Area, Volume: Double;
  RadiusStr: String;
begin
  RadiusStr := '1.0';
  if InputQuery('Trigonometry 101', 'Enter a radius', RadiusStr)
  then
  try
    Radius := StrToFloat(RadiusStr);
    Area := Pi * Sqr(Radius);
    Volume := Pi * Power(Radius, 3) * 4/3;
    ShowMessageFmt(
      'A circle with radius %g has area %g'#10 +
      'A sphere with radius %g has volume %g'#10 +
      'Calculation performed at %s on %s',
      [Radius, Area, Radius, Volume,
      TimeToStr(Time), DateToStr(Date)])
  except
    on EConvertError do
      MessageDlg(
        'Your entry was not recognised as a valid number',
        mtInformation, [mbOk], 0)
    end
  end;
end;
```

If any code in the try block generates an **EConvertError** exception (in truth the **StrToFloat** call is the only likely line) then execution immediately jumps to the **EConvertError** handler. Neither the calculation statements nor the call to **ShowMessageFmt** need to be incorporated into a condition that checks if the floating point number was valid, since they would not be executed if this were the case (they would be skipped en route to the exception handler).

The exception handler in this case simply displays its own error message box containing a different string (see **figure 10**). But again, you can execute any appropriate code that you want to here.

Note that you can handle as many exceptions as you choose by writing multiple exception handlers:

```
try
```

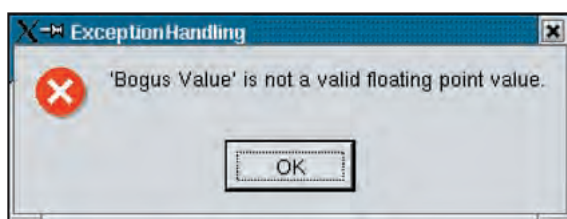


Figure 9: The default EConvertError message.



Figure 10: A Different Response To EConvertError.



```

<< //code that may cause exceptions
except
  on ExceptionType1 do
    // ExceptionType1 handler
  on ExceptionType2 do
    // ExceptionType2 handler
  //and so on
end

```

An Exception And How To Get It

The **Exception** class implements just one useful property for exception handlers to use. The **Message** property returns the string displayed in the default exception handler's message box.

Some exception classes expose a little more information that can prove useful when writing an exception handler. One example is the **EInOutError** exception that's generated for file I/O errors when the appropriate compiler switch is enabled. The **I/O Checking** switch on the Compiler page of the project options dialog controls this and it is enabled by default.

When this option is enabled and an I/O error occurs, such as access being denied or a file not being found, an **EInOutError** exception is raised and its **ErrorCode** field (not a property but a data field) is set to an appropriate error code. If the value is between 0 and 99 it represents an OS-specific error (the meanings often differ between Linux and Windows) and if it is between 100 and 149 it represents a platform-independent error generated by **BaseCLX**. You can find more about these errors in the help under *EInOutError* or *I/O errors*.

If you need to access anything in an exception object the syntax used so far is a bit limiting. It tells you that the exception is of the specified class type but does not allow access to the exception object. An optional extension to the syntax overcomes this small issue, as in:

```

try
  //code that may cause an exception
except
  on E: Exception
    //exception handling code
end

```

For the duration of the exception handler the exception object is available as **E** (although any valid identifier can be used), so **E.Message** tells you the exception description.

Some text file manipulation code that uses the I/O error code is shown below. The code assumes an existing file is chosen in a dialog in order to have a simple line of text appended to it.

```

uses
  Libc;
...
procedure TForm1.Button2Click(Sender: TObject);
var
  TF: TextFile;
begin
  if dlgSave.Execute then
  try
    AssignFile(TF, dlgSave.FileName);
    Append(TF);
    WriteLn(TF, 'Hello world');
    CloseFile(TF)
  except
    on E: EInOutError do
      case E.ErrorCode of
        ENOENT {2}: ShowMessage('"%s" cannot be located'
          , [dlgSave.FileName]);
        EACCES {13}: ShowMessage('You do not have

```

```

permission to overwrite this file');
      else
        if E.ErrorCode < 100 then
          ShowMessage('I/O Error'#10#9'Error code:
            %d'#10#9 +
              'Error msg: %s', [E.ErrorCode,
                SysErrorMessage(E.ErrorCode)])
        else
          ShowMessage('I/O Error'#10#9'Error code:
            %d'#10#9 +
              'Error msg: %s', [E.ErrorCode, E.Message]);
        end; //case
      end //try
    end; //event handler

```

The **TF** variable is used to access the text file whose name is chosen using a **TSaveDialog** component. If any I/O error occurs an **EInOutError** exception is raised automatically and so the **EInOutError** exception handler executes.

The exception handler has a specific response for error codes **2** and **13** (represented by the **ENOENT** and **EACCES** constants from the *Libc* unit) providing user-friendly messages. However for all other error codes a generic message is displayed.

For OS errors, the **SysErrorMessage BaseCLX** routine is used to get a descriptive message to display along with the error code (**figure 11** shows what happens when you choose your own running executable file). For other error codes the message that came with the exception is coupled with the error code. Both these generic messages (and one of the previous messages) are split over several lines using control codes (**#10** is a line feed character and **#9** is a tab character).

Note that, as mentioned before, the error codes below 100 are OS-dependent. The constants used in the case are defined in the *Libc* unit. In a *Delphi* application the corresponding OS error constants are defined in the Windows unit (**ERROR_FILE_NOT_FOUND** has a value of **2** and **ERROR_ACCESS_DENIED** is **5**). You could use conditional compilation directives to write code that is portable between the two platforms as shown below. We will doubtless look further at conditional compilation in some future instalment.

```

case E.ErrorCode of
  {$ifdef LINUX}ENOENT{$endif}
  {$ifdef MSWINDOWS}ERROR_FILE_NOT_FOUND{$endif}:
    ShowMessage('"%s" cannot be located', [dlgSave.FileName]);
  {$ifdef LINUX}EACCES{$endif}
  {$ifdef MSWINDOWS}ERROR_ACCESS_DENIED{$endif}:
    ShowMessage('You do not have permission to overwrite this
    file');
  else
    ...
end;

```

Locating Exception Handlers

When an exception occurs in a *Kylx* application the closest exception handler to it is executed. That means that if the offending statement is in the try block of a **try..except** statement that contains an exception handler for that exception type, it will be executed.

If not, the routine that called the current one is checked for an exception handler that contains the call. If it has no luck there the search continues back to the caller of that routine, and so on as long as there are nested calls. At some point an exception handler will be found or the top most routine will have been checked. If no exception handling statement is found then the application's default exception handler takes over and handles

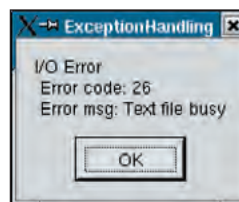


Figure 11: A custom response to a specific I/O error.

the exception as we saw earlier.

If you think about these mechanics you can see that in a complex application with many cases of routines calling other routines there will be cases where an exception occurs in one routine and the exception handler is found several functions further up the function call stack. In order for this to work properly all the intermediate routines (including the one that triggered the exception) have to be voided. It is not possible to return back to where the exception originated from. *Kyl* therefore implements what is called a *non-resumable exception model*.

Chaining Exception Handlers

Let's say you have routine A and routine B. Routine A calls routine B and B contains a variety of statements. Due to the possibility of B causing an exception the implementation of A contains a **try..except** handler to handle the exceptions in some way. Maybe the exception handler logs details of the exceptions to a log file or performs some other useful handling task. The code below shows the picture we are trying to paint:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  //more code
  A
  //more code
end;

procedure TForm1.A;
begin
  try
    //more code
    B
    //more code
  except
    on E: EConvertError do
      begin
        ShowMessage('We just logged a problem: '#10#10 +
          'A %s exception occurred, saying: '#10'"%s",
          [E.ClassName, E.Message])
      end
    end
  end;
end;

procedure TForm1.B;
var
  Radius, Area: Double;
  RadiusStr: String;
begin
  RadiusStr := '1.0';
  if InputQuery('Trigonometry 101', 'Enter a radius', RadiusStr)
  then
    begin
      Radius := StrToFloat(RadiusStr);
      Area := Pi * Sqr(Radius);
      ShowMessageFmt('A circle with radius %g has area %g',
        [Radius, Area])
    end
  end;
end;
```

Now consider what happens if you come back to routine B some time later and decide to write an exception handler in there to perform some necessary exception handling, perhaps to produce a more user-friendly error message. The result of this is that the exception handling code in A will now stop working as no exceptions are being generated by B that are

left unhandled.

```
procedure TForm1.B;
var
  Radius, Area: Double;
  RadiusStr: String;
begin
  RadiusStr := '1.0';
  if InputQuery('Trigonometry 101', 'Enter a radius', RadiusStr)
  then
    try
      Radius := StrToFloat(RadiusStr);
      Area := Pi * Sqr(Radius);
      ShowMessageFmt('A circle with radius %g has area %g',
        [Radius, Area])
    except
      //This kills off A's exception handler - it will never be
      triggered now
      on EConvertError do
        MessageDlg('Your entry was not recognised as a valid
        number',
          mtError, [mbOk], 0);
    end
  end;
end;
```

This might not be the effect you want so there is a mechanism available through the reserved word **raise** to regenerate the same exception in your exception handler as you just caught. This causes the execution to jump to the next closest exception handler. If **raise** is executed in the exception handler in B it will chain onto the exception handler in A.

```
procedure TForm1.B;
var
  Radius, Area: Double;
  RadiusStr: String;
begin
  RadiusStr := '1.0';
  if InputQuery('Trigonometry 101', 'Enter a radius', RadiusStr)
  then
    try
      Radius := StrToFloat(RadiusStr);
      Area := Pi * Sqr(Radius);
      ShowMessageFmt('A circle with radius %g has area %g',
        [Radius, Area])
    except
      on EConvertError do
        begin
          MessageDlg('Your entry was not recognised as a valid
          number',
            mtError, [mbOk], 0);
          //This chains back to the next closest exception handler
          raise
        end
      end
    end;
end;
```

In the case of the example code, if an invalid number is entered the error shown in **figure 10** is displayed followed by a message confirming that the logging code is executing.

Summary

Next month we'll continue looking at exception handling and related issues, including application-wide exception handlers, raising exceptions, custom exceptions and resource protection. In the meantime, if there is something about *Kyl Open Edition* you want to see covered here, drop us an email and we'll try our best to incorporate it into a future instalment. [LXF](mailto:brian@blong.com)

About Brian Long

Brian Long is a UK-based freelance trainer and problem solver for Borland's *Kyl*, *Delphi* and *C++Builder* packages. His Web site is at www.blong.com and he can be emailed at brian@blong.com



VIRTUAL PRIVATE NETWORKS

Building Linux VPN tunnels

Dan DiNicolo explains how to use Linux running FreeS/WAN to build an effective IPSec VPN tunnelling solution.

The best thing about Linux is that you can make it do just about anything. For many of you, Linux is possibly already your router, firewall, NAT server, and more. With a little work, you can easily extend your Linux setup to build a cost-effective WAN replacement in the form of IPSec VPN tunnels.

When considering a VPN solution, it's usually important to insist on two things. The first is that it supports at least end-to-end tunnels and roaming users, the second being that it should be standards-based (that means IPSec support) to make interoperability with other systems possible when required. This article covers how to install and configure *FreeS/WAN* to create secure, IPSec-based VPN tunnels between locations. If your needs are relatively simple, you'll have a secure Linux-based VP solution securing your traffic between locations in no time. If you want to get fancy, extending the design outlined here isn't much more difficult at all.

The main purpose of deploying an IPSec-based VPN tunnel solution is as a replacement or backup for your current (and probably expensive) WAN links. Companies spend thousand of dollars on WAN infrastructure that can potentially be avoided with the proper implementation of a VPN over a standard (but hopefully high-speed) Internet link. The idea is to use Linux and an unsecured network (such as the Internet) as a vehicle for secure communications between two or more locations. This will provide easy and seamless connections to remote network servers and resources for our local network users.

This tutorial assumes the use of Red Hat 7.2, on which *FreeS/WAN* implementations tend to go smoothly. To that end, just about any other distribution of Linux with the appropriate config should work with a few modifications to the network startup parameters, which may be configured differently.

Understanding your tunnels

Before getting into the step-by-step instructions, it's important to understand what you're trying to configure, and the technologies involved. Essentially the goal is to create a Linux gateway at each location that will secure traffic that passes between the locations using IPSec. These gateways might also be our router, NAT server or firewall. What our *FreeS/WAN* implementation will do is watch for network traffic destined for the remote network, encrypt it using the IPSec Encapsulating Security Payload (ESP) protocol, and use the Internet as a vehicle for its transmission. On the receiving end, the *FreeS/WAN* system will decrypt the packets and forward them on to the designated subnet we have defined in our configuration files. Note that the interior systems need not know about or understand the encryption process; they simply attempt to communicate as normal. **Figure 1** provides a high-level overview of the network configuration we'll be using.

To begin with, you'll need at least two servers with Linux installed, assumed to be Red Hat 7.2. You might consider PII's with at least 128 MB of RAM. Remember that a great deal of encryption is computationally taxing, so use what you can afford. To that end, you can also easily install and run *FreeS/WAN* on lesser boxes if that's all you have – testing in your particular environment will show whether the performance meets your needs. You'll also need at least two network cards or one network card and some type of PPP connection in each server if that's the route your taking.

Getting ready to install

Before you install *FreeS/WAN*, a number of packages should be on hand or installed. It is imperative that you have the kernel source files, a GNU C compiler (probably *gcc*), as well as *make* and *patch* installed. Other required files include the *GMP* library, which will be required for public key calculations. According to the developers, not installing the *GMP* library and *patch* are two of the most common mistakes, so ensure they have been installed. If you are using Red Hat 7.2, take the time to query for the

following packages, and install any not found using **rpm -i**

```
rpm -q gmp-3.1.1-4.i386.rpm
rpm -q gmp-devel-3.1.1-4.i386.rpm
rpm -q patch-2.5.4-10.i386.rpm
rpm -q make-3.79.1-8.i386.rpm
```

You'll also need the *FreeS/WAN* source files, which can be downloaded from <http://www.freeswan.org>. The current version is 1.94, and is not included as an RPM with Red Hat 7.2. Download the file to `/usr/src` and untar it there:

```
tar -xzf freeswan.1.94.gz ***
```

Beginning the *FreeS/WAN* installation is relatively simple and painless, as most of the installation is scripted. It is strongly recommended that you first build your own kernel *prior* to the installation, as this will help if a problem needs to be tracked down later. For the purpose of building this system, go through a kernel configuration and create a new image for *FreeS/WAN* using the default 2.4.7-10 kernel. If you plan to use an earlier kernel version, the developers point out that *FreeS/WAN* scripts will not compile properly on the 2.2.19 series kernel – you should use 2.2.20 at a minimum. Note also that *FreeS/WAN* will ultimately look for the Linux kernel source in the `/usr/src/linux` directory, so create a symbolic link prior to starting:

```
ln -s linux-2.4 linux
```

Installing the new kernel and image should be straightforward; the amount of time it takes will depend on the hardware you have chosen. If you choose to install on Red Hat 7.1 instead, precede this command with a **make mrproper** to avoid script errors. The standard options to configure the kernel include:

```
Make menuconfig
Make dep
Make bzImage
Make install
Make modules
Make modules_install
```

After the processes have completed, copy the new bzImage file to `/boot`.

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot
```

You'll then need to edit `/etc/lilo.conf` to point to the new image. Use the current boot loader settings found and create a new path for an alternate image called IPsec. For the sake of keeping things simple, make this the default kernel to boot from. It isn't very hard to accidentally boot to the wrong image and then wonder why your configuration isn't working. Remember to run *lilo* after saving the file to be sure it sees the changes you've made. After the process has completed, reboot your system using the new image.

The actual build of *FreeS/WAN* can take many forms. You can build it to load as a module, install directly from the source, and, as of version 1.93 you can even build your own RPMs if you choose. We'll be installing from source, but building RPMs might be a worthwhile undertaking, especially if you plan to deploy a large number of gateways. The installation from source involves two major processes. The first reconfigures and recompiles the kernel to support IPsec, while the second actually installs the new kernel. There are a number of configuration options available if you're interested in controlling the actual IPsec options, but by choosing the fully automated versions you'll save time, and they work exceptionally well.

To reconfigure and recompile the kernel, issue the following from the `/usr/src/freeswan-1.94` directory:

```
make oldgo
```

You should note that depending on your hardware, this process might take a while. Since an RSA authentication key pair is also generated at this point, and relies heavily on the use of

random, you might consider providing input by moving the mouse or typing on the keyboard during the process. If that bores you to tears, considering using Alt+F2 to open a new terminal session and run **du /usr > /dev/null** to generate disk activity. If the very last line of *make's* output is as follows – `utils/errcheck.out.kbuild` – you'll know the process completed successfully. If not, you will need to check the `out.kpatch` and `out.kbuild` files for details.

After the kernel has been recompiled, the updated kernel needs to be installed. This process has been automated as well. From the `/usr/src/freeswan-1.94` directory enter the following:

```
Make kinstall
```

The last line of the script should call another `errcheck` script signalling that the install was successful – `utils/errcheck.out.kinstall`. Finally, check that your kernel configuration is correct in `lilo.conf`, and run *lilo* again. Now reboot into your newly compiled and installed kernel

If the installation has completed and is loading successfully, you should see the ipsec start-up message at boot time. You can also issue the following commands from the prompt to get more information, or view running processes using **ps -ax|less** and scroll through the list.

```
ipsec --version
```

```
ipsec whack --status
```

Configuring Free S/WAN

The configuration of *FreeS/WAN* is not terribly difficult but can be a little bit tricky. You will need to configure some general start-up parameters and create connection objects that define the tunnels you wish to implement. To start, draw a basic diagram of the implementation scenario, which can be added to later. There are many variables that need to be documented at this point including the interfaces and their IP addresses, private subnet addresses, and the address of the next-hop Internet gateways. For this example, assume that what we're trying to connect are two single subnet networks that are connected to the Internet using either straight routing or masquerade. On the test network, we're using masquerade to NAT private internal IP addresses (192.168.x.y) to public external interfaces, with this configured correctly on both gateway systems running *FreeS/WAN* (note that for illustration purposes, the entire test network is running private addresses). Internal clients should ultimately point to the IPsec server's internal interface as their default gateway. Given that IPsec isn't configured with any tunnels yet, we'll assume that your private internal clients can ping Internet systems from both networks. Common errors at this point mainly involve forgetting to enable the required IP forwarding on the servers. If you haven't already done this, issue the following command from the prompt for testing purposes:

```
Echo "1" > /proc/sys/net/ipv4/ip_forward
```

To enable IP forwarding automatically at startup on a Red Hat system add the following command to `/etc/sysconfig/network`:

```
FORWARD_IPV4=true
```

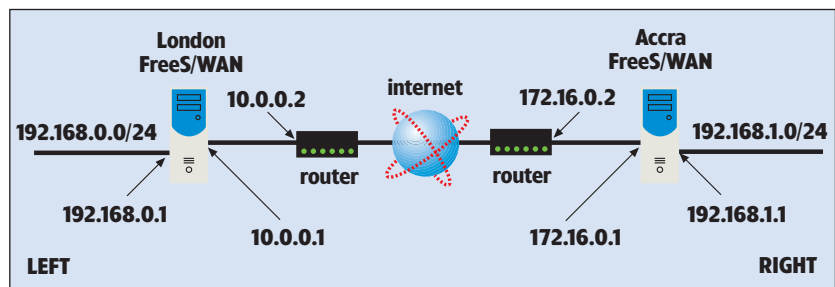


Figure 1: Seamless and secure connection for your WAN.

LinuxFormatTutorialVPN

◀ If your configuration uses real IP addresses and routing on both networks, you should already be able to ping systems on the remote LAN without issue. Testing to ensure that the network is communicating correctly prior to configuring *FreeS/WAN* is critical – isolating the problem source will be difficult if you don't have this sorted out to begin with.

The two main configuration files that you'll be dealing with are *ipsec.conf* and *ipsec.secrets*, both found in your */etc* directory. The *ipsec.conf* file contains the service start-up parameters, as well as the connection definitions for the tunnels. The *ipsec.secrets* file contains the private and public RSA key pair that was generated automatically during installation and will be described shortly.

Getting back to that diagram mentioned earlier, you should designate one side 'Left' and the other side 'Right'. It doesn't actually matter which is which, as long as the configuration is consistent. Each side should also have a real name based on something like location. In our example, we're going to build a tunnel from London to Accra, so we'll call the connection *london-accra*, with London being designated the Left side and Accra the

Figure 2: the connections are defined in *ipsec.conf*

```
/etc/ipsec.conf - FreeS/WAN IPsec configuration file

# More elaborate and more varied sample configurations can be found
# in FreeS/WAN's doc/examples file, and in the HTML documentation.

# basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work;
    # %defaultroute is okay for most simple cases.
    interfaces=%defaultroute
    # Debug-logging controls: "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoauto=search
    plutoauto=start
    # Close down old connection when new one using same ID shows up.
    uniqueids=yes

# defaults for subsequent connection descriptions
# (mostly to fix internal defaults which, in retrospect, were badly chosen)
conn %default
    keyingtries=0
    disablearrivalcheck=no
    authby=rsasig
    lefttsasigkey=%dns
    righttsasigkey=%dns

# connection description for (experimental!) opportunistic encryption
# (requires KEY record in your DNS reverse map; see doc/opportunism.howto)
conn me-to-anyone
    left=%defaultroute
    right=%opportunistic
    keylife=1h
    rekey=no
    # uncomment to enable incoming; change to auto=route for outgoing
    #auto=add

# sample VPN connection
conn sample
    # Left security gateway, subnet behind it, next hop toward right.
    left=10.0.0.1
    leftsubnet=172.16.0.0/24
    leftnexthop=10.22.33.44
    # Right security gateway, subnet behind it, next hop toward left.
    right=10.12.12.1
    rightsubnet=192.168.0.0/24
    rightnexthop=10.101.102.103
    # To authorize this connection, but not actually start it, at startup.
    # uncomment this.
    #auto=add
```

Right side, exactly as they are written. Adding this to your diagram will help you immensely during the configuration process.

The first file we'll work with is *ipsec.secrets*. On each *FreeS/WAN* server, this file will contain the server's public and private key. This key pair is used for authentication purposes, with the only other option being a relatively weak shared secret. While you don't need to worry about the public part of the key, you definitely should be worried about the protection of its private counterpart. The public part of the key will need to be extracted and ultimately be put into the *ipsec.conf* file. To do this, I suggest extracting the public key to a separate file on each server, and then setting restrictive permissions on *ipsec.secrets*. Thankfully, *FreeS/Wan* provides a very simple way to extract the public portion of the key in the correct format. Assuming that you are working from the London server (which is designated Left), type:

```
ipsec showhostkey --left > leftpubkey.txt
```

This will output the London (or Left) public key to a new file called *leftpubkey.txt*. Do the same on the Accra (or Right) server, but use `--right > rightpubkey.txt` instead. After doing that, it is a good idea to *chmod* the *ipsec.secrets* files to **600**.

After the keys are generated, you'll need to have them both on the same server to build the *ipsec.conf* file, and this presents a small dilemma. You will want to be sure that the public key you are receiving from the remote server is the correct one, and has not been created by some malicious third party. If you're creating the servers right next to each other this might not be an issue, but you could be dealing with someone you have never met before or something similar. Creating a secure tunnel with an unknown third party could have huge security consequences, so it is generally recommended that you use PGP to sign public key files (or the email message used to transmit them) to be sure of the identity of the sender. If your servers already communicate over a secure or direct connection, transfer a key file from one server to the other. In this case, FTP the Accra *rightpubkey.txt* file to the London server.

The file that controls how connections are defined and initiated is *ipsec.conf*. If you open this file in *vi* (as shown in **figure 2**), you'll see a number of configuration parameters. The file contains three main sections, each of which is important to understand. The top part of the file describes the machine configuration, in a section called **config setup**. The next section is called **conn default**, and contains settings that will apply to all connections. After these you will find actual connection descriptions, which is where we'll ultimately define our *london-accra* tunnel. Before editing this file, make a backup copy called *ipsec.old* or similar.

The **config setup** section is not difficult to configure because most of the default values will do. However, it is worth knowing the purpose of each variable in case you run into any problems. The **interfaces=%defaultroute** can probably be left alone, since it will assign the default Internet connection interface as the *ipsec0* interface. If not, you can assign the appropriate interface name itself, such as **interfaces="ipsec0=eth0"**. The following two variables, **klipsdebug=none** and **plutodebug=none** turn off debugging for Kernel IPSec (KLIPS) and *Pluto* (the connection negotiation daemon). These can be set to **all** if debugging is required. The **plutoauto=** and **plutoauto=search=** are both set to **%search** by default. This means that the variables configured on the actual connection description will be used to decide whether the connection is simply loaded into memory or negotiated when *Pluto* starts. For all intents and purposes, leave these alone. If set to **yes**, the **uniqueids=** variable will remove old connections in cases where one server closes a connection and then attempts to reconnect.

The **connection defaults** section is where we'll need to add the public keys generated earlier, and is critical for the mutual authentication of our tunnel servers. You'll need to insert the public key variables in the appropriate **lefttrsasigkey=** and **righttrsasigkey=** sections. To do this quickly and easily, open **leftpubkey.txt** in *vi* and move the cursor to the **I** of **lefttrsasigkey=**. In command mode enter the following:

```
yy
:e /etc/ipsec.conf
```

This will copy the key and then open the **ipsec.conf** file to allow you to paste in the new value. Move to the **authby=** line. Enter:

```
p
```

This will paste the key into the file – just be sure to remove the old **lefttrsasigkey=** line, and then follow the same procedure using **rightpubkey.txt**. Be sure that sections are aligned properly for clarity and proper functionality. If your setup involves the creation of multiple tunnels, the public key values should be moved into the actual connection description instead of being left in the default section, as different key values will exist for different tunnels.

You might notice that the next section is a connection description for the purpose of opportunistic encryption, an experimental feature. Since you likely won't be using this, comment out this section or remove it altogether.

The last critical configuration task will be creating our connection description. Consider directly editing the sample as opposed to re-typing everything. You'll need the information from the diagram created earlier to fill in all of the variables. By examining the sample diagram, setting the majority of the connection variables below should be obvious:

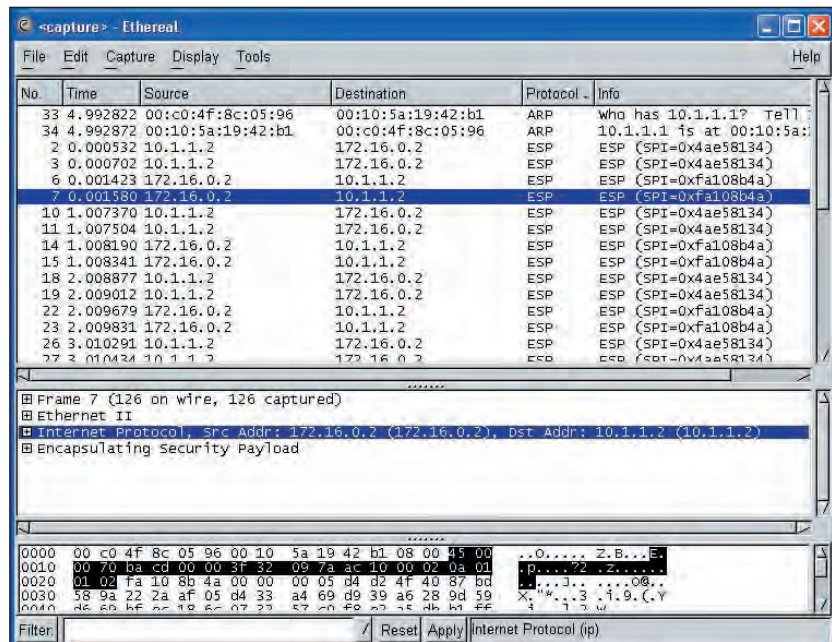
```
conn london-accra
    left=10.0.0.1
    leftsubnet=192.168.0.0/24
    leftnexthop=10.0.0.2
    right=172.16.0.1
    rightsubnet=192.168.1.0/24
    rightnexthop=172.16.0.2
```

For cases where you have multiple subnets behind the gateway, configuration will simply involve changing the subnet mask value in the left or right **subnet=** field. For example, if you had both the 192.168.2.0/24 and 192.168.3.0/24 networks in a location, you could summarize them by changing the mask value to /23. If you are testing in a lab environment without a router, the **nexthop** sections of the connection can be removed, but in production they will be necessary. The final option in this section is a variable called **auto=add**. If uncommented in its current state, the connection will be added to memory when the system boots but will not be started. To have the connection start automatically, change this option to start. For the purpose of testing, uncomment it and leave it set to **add**. After being sure it works, you can change it to start.

After your configuration is complete, remember that this **ipsec.conf** file will need to be transferred to the **/etc** directory on the Accra server, overwriting the default file that still exists there. After this is complete, reboot both servers. After both are again online, and assuming you have left the **auto=** option in **ipsec.conf** set to **add**, you can initiate the connection from one of the systems. For my london-accra connection, the command is:

```
ipsec auto --up london-accra
```

After doing so, you should see a brief negotiation message mentioning that a security association has been established. If not, you've likely made an error in the **ipsec.conf** file. An important note here; while the connection need only be initiated from one of the systems, it should always be shut down on both



systems. If not, it will continue to wait for re-establishment, wasting system resources. To shut down a connection from either server, use the command:

```
ipsec auto --down london-accra
```

To view the currently connected tunnels use:

```
ipsec look
```

After the tunnels have been created, you'll want to be sure that the traffic moving between them is encrypted. One common mistake is simply pinging from one **FreeS/WAN** gateway to the other. This doesn't prove anything, since the only traffic that will be encrypted is that which moves from one subnet to the other. To test properly, you should attempt to make a connection from one subnet to the other and capture the traffic that moves between the two. For example, start a file transfer from a client in Accra to a server in London. To view the actual data stream, set up an intermediate system with a packet capture utility such as *Ethereal*. If data is properly ESP-encrypted, you should see a number of protected packets whose contents cannot be viewed, as it **figure 3**. The source and destination addresses should be the respective gateways.

In order to ensure that any intermediate firewall (or your IPsec gateways configured as a firewall) passes traffic properly, the following rule sets need to be created. This assumes that you're running iptables, but any firewall will allow you to establish the appropriate rules. For *FreeS/WAN*, we'll need a rule that allows UDP port 500 traffic to pass (used for the tunnel negotiation process), and also allows ESP traffic to pass, which uses IP protocol 50. Configuring the rules will involve setting up the following filters:

```
iptables -A INPUT -p udp --sport 500 --dport 500 -j accept
iptables -A OUTPUT -p udp --sport 500 --dport 500 -j accept
iptables -A INPUT -p 50 -j ACCEPT
iptables -A OUTPUT -p 50 -j ACCEPT
```

Once you have your *FreeS/WAN* gateways set up, and firewall parameters properly configured, you're off to the races. The design can be further extended to include multiple tunnels and roaming user configurations. To that end, *FreeS/WAN* is well documented and has a great mailing list where you can easily get all of your questions answered. Just be sure that you've consulted the documentation prior to asking, because you're likely to be referred right back to it. Happy VPNing! **LXF**

Figure 3: Success! The data is safely encrypted.

Linux FreeS/WAN Resources

Where to get FreeS/WAN:
<http://www.freeswan.org>
 A good configuration page:
<http://jixen.tripod.com/>
 The mailing list:
<http://www.freeswan.org/mail.html>

Coverdisc

Once again **Neil Bothwick** has scoured the world of Linux to bring you an essential selection of the best software, the most useful applications and the tools that will best enhance your Linux life. Enjoy...



Important notice

Before you even put the CD or DVD in your drive, please make sure you read, understand and agree to the following: The Linux Format CD/DVD is thoroughly tested for all known viruses, and is independently certified virus-free before duplication. We recommend that you always run a reliable and up-to-date virus-checker on ANY new software. While every care is taken in the selection, testing and installation of CD/DVD software, Future Publishing can accept no responsibility for disruption and/or loss to your data or your computer system which may occur while using this disc, the programs or the data on it. You are strongly advised to have up-to-date, verified backups of all important files. Please read individual licences for usage terms.



Wherever you see this logo it means there's related stuff on the CD

READ ME FIRST

This month we once again bring you a choice of two CDs or a DVD. The first CD and the DVD are bootable and contain the Trustix Linux installer. If your BIOS doesn't support booting from CD, go to the disc's "images" directory in a console, put a blank disk in the floppy drive and type **dd if=boot.img of=/dev/fd0**. Then you can boot from this disk with the CD in the drive. The first CD also contains the Magazine directory, with files relating to the various articles in the magazine.

The second CD contains Sorcerer and the normal CD contents, and is bootable if you want to install Sorcerer. As before, you can make a boot floppy if your computer won't boot from CD. This time, cd to the CDROM's root directory and type **dd if=btmgr.floppy of=/dev/fd0**. This will create a BootManager disk, which can be used to start a Sorcerer install.

Desktop

KAlarm sits in the *KDE panel* and pops up a window when it's time for you to do something. There is also a todo list and the option to run a program as well as, or instead of, giving a reminder.

Staying with popup messages, the Windows SMB protocol has the facility to send popup messages as well as sharing files and printers. *Popper* sends and receives messages for Windows machines using this method.

Development

Spectix helps you do develop GUIs in a graphical environment and then generate code to create the GUI on Unix, Windows and MacOS. Even if you prefer a graphical IDE you will always need to work with source code in an editor. We have two text editors, specifically aimed at programmers, on the CD this month. *Minimum Profit* has a small memory and disk footprint, while *SetEdit* has a wide range of features and is designed for editing multiple files.

Games

Adonthell is a work-in-progress role-playing game engine. We have the

latest version of the engine for you, along with a complete game for it. If you prefer something a little faster and more real time try *Grande*, a 2D scrolling shooting game.

Graphics

This month we have three video programs for you. *Avidemux* is an editor for AVI files, allowing you to work with the audio as well as the video component. *EfecTV* is a real time video effects program, it needs a video source and capture card to be really useful. *Jahshaka* has features from both of the previous programs, providing editing facilities and real time effects processing.

The Graphics directory also contains several other programs for completely different tasks, such as CAD and presentation software and an interesting program that attempts to categorise the content of images.

Internet

Animail is described as a "mail retrieval utility" – but it does much more than that. Rather than simply downloading mail from your ISP and storing it in your local mailbox, it also provides advanced filtering facilities, which

Essential info

Missing something?

As many of the programs on our discs are the very latest releases, they are often built on the very latest libraries and may depend on other packages your current Linux setup does not contain. We try to provide you with as many of these important supporting files and libraries as possible, though obviously we don't have space to include absolutely everything.

In many cases the latest libraries and

other packages you might need will be included in the "essentials" folder on the disc, so if you are missing dependencies, this is the first place to look.

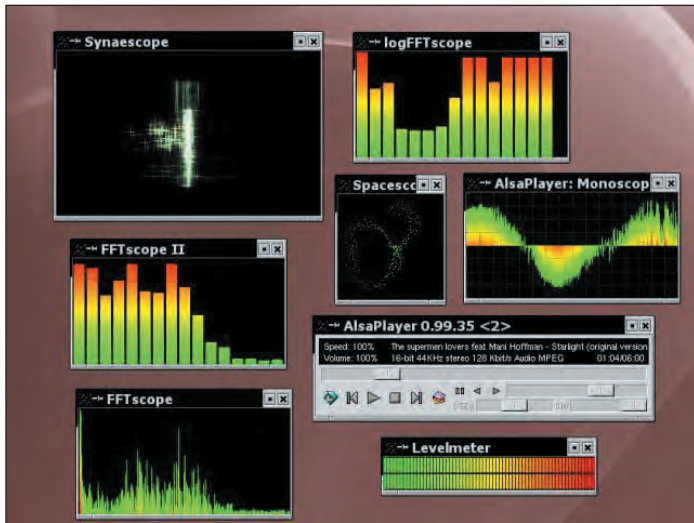
Package formats

Wherever possible, we try to include as many different types of package for an installation as possible, whether that be distribution specific RPMs, debs or whatever. Please bear in mind that we can only do this where space permits and when the packages are available.

We will, apart from exceptional or legally restricted situations, include the source files for any package, so that you can build it yourself.

Documentation

These pages provide helpful information on how to install and use some of the packages on the CD. Please note that many of the applications come with their own documentation, and there are additional notes and files in the relevant directories.



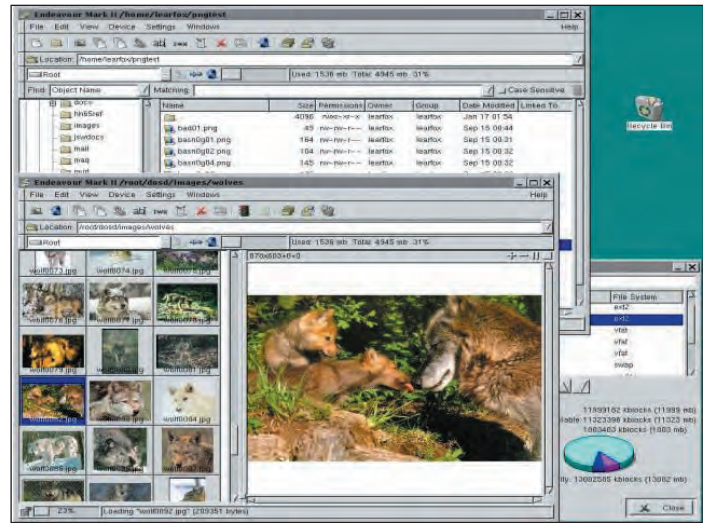
An audio player for the ALSA sound system.

should enable you to reduce the amount of unwanted mail you receive.

So much of the Internet software we see now is aimed at filtering spam and viruses, but it's not all gloom. *HTTrack* is a useful tool that downloads part or all of a web site to your hard drive. It's especially useful if you have a relatively slow Internet connection as, once grabbed, the pages and images load at hard drive speeds.

Office

Time management is always important in an office, and if you're freelance it's vital to know how much you've spent on each project. *KTimeClock* is a KDE program to record how much time you spend on individual tasks and sub-tasks. Nesting of sub tasks means you can also break down the time spent on each part of an individual project.



Endeavour II is a file manager with built in image and archive browsers.

As well as recording what you spend your time on, you usually need to keep track of what how spend your money. As its name implies, *Tiny Business Accounting* is aimed at the smallest businesses. *CALAMAR* is aimed a step higher, at small companies.

Server

Apache 1.3 has been around for years. Its successor, *Apache 2*, has been in development for a while, as evidenced by the fact that the beta version we have for you here is the 32nd beta. We also have a couple of programs designed for small networks. *WWWOFLE* is a web proxy intended for use over dialup connections, it has useful features like queuing requests when offline and fetching them when you next dial up. *Dnsmaq* is a small caching nameserver to serve the local network and pass other requests to your ISP's nameservers.

Sound

MUSE is a program to mix and encode audio streams from a variety of sources and either play the result locally or sent the output to a network streamer. The *Lyric Display System* uses a second display, either a monitor or video projector, to edit or display song lyrics. It was written for use at church services, but will probably find itself in use at a few karaoke parties. If you prefer a more private appreciation of music, *RioUtil* lets you transfer songs to and from Diamond Rio MP3 players, as well as performing other tasks like firmware upgrades.

What are all these files?

If you are new to Linux, you may find the profusion of different files and extensions confusing. As we try to give as many packages as possible for compatibility, there will often be two or three files in a directory covering different types of Linux, different architectures and usually source and binary versions – so which do you install? They can be identified by their filenames, and usually just by the file extensions.

Someap-1.0.1.i386.rpm – This is probably a binary rpm, designed to run on x86 systems.

Someap-1.0.1.i386.deb – The same, but a debian package.

Someap-1.0.1.tar.gz – This is usually source code.

Someap-1.0.1.tgz – Same as the above, tgz is abbreviated form of tar.gz

Someap-1.0.1.tar.bz2 – Same, but uses bzip2 compression instead of zip

Someap-1.0.1.src.rpm – This is also source code, but supplied as an rpm to make it easier to install

Someap-1.0.1.i386.RH7.RPM – A binary, x86 RPM designed specifically for Red Hat Linux

Someap-1.0.1.ppc.Suse7.rpm – A binary RPM designed specifically for SuSE7.x PPC Linux.

Someap-devel-1.0.1.i386.rpm – A development version.

System

Printing is one of those things that works well for some people and causes no end of hassles for others. *CUPS* is becoming the standard printing system and we have the latest version for you here. To go with it,

Creating install CDs

The quickest way to burn an ISO image to CD is with *cdrecord*. You need to be root to do this. First find the address of your CD writer with

```
cdrecord -scanbus
```

This will show the devices connected to your system. The SCSI address of each device is the three numbers in the leftmost column, say 0,3,0. Now you can burn a CD with

```
cdrecord dev=0,3,0 -v
/path/to/image.iso
```

You can simplify the command by saving some default settings in */etc/default/cdrecord*. Add a line for each CD writer on your system (usually one) like this

```
Plextor= 0,3,0 12 16M
```

The first item is a label, after the SCSI address you put the speed and the buffer size to use. You can now replace the SCSI address in the command line with the label, but it gets even easier if you add

```
CDR_DEVICE=Plextor
```

Now you can burn an ISO image to disc with

```
cdrecord -v
/path/to/image.iso
```

If you really don't want to use the

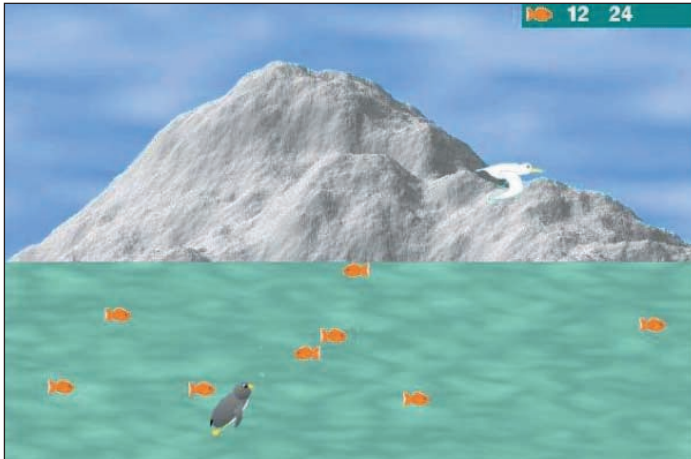
command line, *gcombust* will do the job for you. Start it as root, select the "Burn" tab and the "ISO 9660 Image" gadget near the top of the window. Put the path to the image file in the gadget and press "Combust!". Now put on the kettle while the CD is created for you.

Other OS?

You do not have to use Linux to burn the ISO to a disc. All the Linux-specific bits are already built into the image file. Programs like *cdrecord* simply dump it to the disk. If you don't have a CD writer, find someone who does have one, and a DVD drive, and use the CD burning software on their computer. It can be Windows, MacOS, AmigaOS whatever.

No CD burner?

What if you have no CD writer? Do you know someone else with one? You don't have to use Linux to burn the CDs, any operating system that can run a CD writer will do the job. The details will vary according to the software used, but any friend with a computer that has both DVD-ROM and CD-R drives is worth buying a couple of drinks to transfer the ISO images to CD for you.



It wouldn't be a Linux coverdisc without a penguin game.

« *gimp-print* is a collection of high quality printer drivers suitable for use with *CUPS*, *GhostScript* and the *GIMP*.

Finding files on your own computer is easy with *find* and *locate*, but what about other computers in the

network? *UntzUntzLANScan* builds a database of files on *Samba* shares, much like *locate* does for your local hard drive, and can be configured as to which shares or directories are included or excluded.

CD and DVD structure

The CDs and DVD use the same layout. These are the directories they are divided into and the type of software each one contains:

Magazine

This contains programs and other files mentioned in various articles in each month's magazine. Look here for versions (maybe demos) of reviewed software or programs referred to in tutorials and features.

Desktop

A wide range of programs for

general desktop usage of a Linux machine, from full blown windowing environments like *KDE* to small utilities.

Development

Anything relating to software development. This includes compilers, libraries, classes, debugging tools and development environments.

Distros

As you would expect, various Linux distributions. These can be full distributions, that are only on the DVD or mini-distros that can fit on a CD.

» CD CONTENTS AT A GLANCE

Magazine (on the first CD)

Apache	The current version of the Apache web server
CVS	CVS as covered this month
Roundup	The DVD players from the Roundup
Emulators	Files mentioned in this month's Emulation article
FreeSWAN	FreeS/WAN, for the VPN tutorial
HotPicks	The programs covered in this month's HotPicks section
Kylix	Kylix2 Open Edition, to use with our tutorial series
LTSP	All you need to go with the LTSP feature
Perl	The files used in this month's Perl tutorial
Security	Various security and firewall programs
Desktop	
AMOR	Amusing Misuse of Resources
bzip2	Portable, lossless data compressor
CDShare	Automatically share CDs via Samba
Eddi	Text editor with macros and syntax highlighting
EndeavourMkII	File browser/manager with image browser and archiver
FileRoller	Archive manager for the GNOME environment
FTX	Graphical text editor for URL-encoded text files
GKrellIM	Stacked monitor charts CPUs, disk, load, memory etc.
GKrellIMLaunch	GKrellIM plugin for one-click access to applications
GKrellIXkb	Xkb keyboard switcher plugin for GKrellIM
GNUstep	Desktop based on the OpenStep standard developed by NeXT
KAlarm	Configure messages to be displayed at scheduled times
muCommander	File manager with a Norton Commander-style interface
MyMiggy	Easy configuration of UAE
Popper	Messaging between Linux and Windows-based machines
RelataSync	Synchronizes Palm Address, Date and ToDo with Relata
SwissDB	File and CD cataloging program similar to Gtkatalog
TeXmacs	Scientific text editor, inspired by TeX and GNU Emacs
TVProg	TV guide software running as a Web server application
VICE	Emulation of the Commodore C64, C128 and VIC20
XScreenSaver	Modular, customisable screen saver and locker for X

Development

ApplC++Development	
Library	Library of useful programming tools written in C++
C++DebuggingSupport	
Library	Well-documented library for debugging C++ applications

DatabaseAbstraction

Layer	Database-independent abstraction layer in C
FAM	Notifies when specific files or directories are changed
LibCGI	Library to assist in the making of CGI applications in C
libcurl	Transfers files using HTTP, HTTPS, FTP, FTPS, etc.
libglade	Library to load a user interface from an XML description
libpop3	Add POP3 functionality to your projects with no hassle
libxslt	C library for GNOME to work with XSLT
MinimumProfit	Programmer's editor with low memory/disk requirements
NetBeansXMLProject	XML editor for the NetBeans 3.3.1 IDE platform
OGRE	Library for games and demos utilising 3D hardware
PLplot	Library of C functions for making scientific plots
pygame	Python extension modules designed for writing games
QuickAndDirtyGameDev	Quick and Dirty Game Development Framework
RamDiskStuffUtility	Inserts files into a RAM disk image inside a boot image
ReqTools	Create requester windows in graphic applications under X
Setedit	Text editor specially designed for programmers
Spectix	Build programs with GUIs that run on multiple platforms
Syslinux	Boot loader for Linux that operates off of MS DOS floppies

Games

Abuse-SDL	A port of the classic Crack-Dot-Com game to SDL
Adontheil	An open source rôle-playing game
Atomix	A little mind game for GNOME
dopewars	Unix rewrite of the MS-DOS program of the same name
Grande	"ZANAC"-esque 2D scrolling shooting game
Marbles	Create a figure out of marbles within a time limit
PengSwim	Game of swimming and eating
RoboCupSoccer	
Simulator	Two teams and coaches interact in simulated soccer game

Graphics

avidemux	Graphical tool to edit AVIs, uses divx4linux and GTK/GDK
DFBPoint	Presentation program using DirectFB and XML files
EffectV	Real-time video effector
EyeFract	Julia and Mandelbrot set (fractal) generator
FREEdraft	FREEdraft is a 2D mechanical CAD program
Jahshaka	Audio and video editing and effects system
Moron	Classify given images based on their content
PhotoScript	Online photo album with thumbnail generation

Internet

5pstats	Stores statistics about pppd usage in a MySQL database
ActivePHPBookmarks	Web-based program to store and display bookmarks

Games

What more is there to say, the Games section contains... well, games.

Graphics

Paint, image processing, movie players, video capture. Anything to do with capturing, manipulating or viewing graphics can be found here.

Internet

The Internet directory contains client side internet and network software. This include the web browsers and email clients, as well as some more arcane Internet tools.

Office

Various types of productivity software are included here. This is mainly word processors, spreadsheets, databases and accounting software, but there are other programs that also fit here.

Server

Covers everything involved in providing services over the Internet or a network, as opposed to using them.

Sound

Anything to do with capturing, playing, processing or converting sound or music will go in here.

System

This is where you will find updates and enhancements for various system components as well as various utilities to improve your usage of your system.

Essentials

The Essentials directory contains various programs and packages that are often required by items on LXF coverdiscs but aren't always included as standard with distributions. It also contains the latest kernel sources.

Installing from tarballs

A tar ball is a two stage archive. First the files are archived into a single file with *tar* and then compressed with *Gzip* or *Bzip2*. To unpack, **cd** to the directory you want to unpack it, usually your home directory and type one of the following two lines:

```
tar xzvf /mnt/cdrom/Desktop/progname/progname-2.1.0.tgz
tar xjvf /mnt/cdrom/Desktop/progname/progname-2.1.0.tar.bz2
```

Use the first for Gzipped files, those ending in .tar.gz or .tgz, and the second for Bzipped files, ending in .tar.bz2 or .tbz2. Naturally, you change the paths to suit the location and name of the archive. This normally unpacks the archive into a directory of the same name, enter that directory with:

```
cd progname-2.1.0
```

To compile and install the software, type the following three commands:

```
./configure
make
su -c "make install"
```

The last line will prompt you for the root password, as this stage must be run as root. If you are already logged in as root, just type **make install**. This will give you a default installation. If you want to change any aspect of the install, type **./configure --help** to see the options available. For example, you are usually able to change the default location with the **PREFIX** argument. When you have finished installing, you may remove the source files with:

```
cd ..
rm -fr progname-2.1.0
```

You should also log out as root, before you do anything you may later regret.

Animail
Axel
BlackHole
Circle
ConnMon
dailystrips
dnstrace
DownloaderForX
ElysiumDownload
Manager
Evolution
Getleft
HTTrack
Opera

Office

AbiWord
BigFacelessReport
Generator
CALAMAR
HTMLDOC
iSQL-Viewer
JAdvisor
KTimeclock
Moregroupware
SAXON
TinyBusinessAccounting

Server

Apache2
Bricolage
Cocoon
dnsmasq
eprints
ht://Dig
ImlugRemoteBackup
JiveForumsXMLdbDriver
Kismet
Load
ModSurvey
mod_layout
mod_ruby
mod_SQLInclude
mod_ssl

Multiserver POP3/APOP/IMAP4 mail retrieval utility
Accelerate downloads by using multiple connections
Spam blocker for use with QMail
Peer-to-peer application for file sharing and messaging
Connection and bandwidth monitoring
Download your favorite online comic strips each day
Determines where a given DNS gets its information
Downloads files from the Internet via both HTTP and FTP

Extensible download manager for the GNOME
Mailer, calendar, contact manager and communication tool
Given a URL, Getleft will try to download all links
Download a web site to a local directory
The first beta of Opera 6

Cross-platform open source word processor

Java component for creating PDF reports from XML
Accounting program entirely written in Java
Converts HTML into indexed HTML, PostScript and PDF
IndependentSQL-Viewer
Class scheduler, course planner and search program
Record the amount of time you've spent on various tasks
Groupware with calendar, news, contacts and more
A collection of tools for processing XML documents
Minimal expense tracking, billing and invoicing

The latest beta of the world's most popular web server
Content-management and publishing system
XML publishing framework from Apache
Provides DNS services to a small network
Solution for sites wanting to set up archives
WWW indexing and searching for a domain or intranet
Sends a copy of a directory to a remote machine
Run Jive discussion forums on top of an XML database
802.11b network sniffer and network dissector
Web services performance and scalability testing
Allow users to create their own web questionnaires
Gives Apache module both Footer and Header directives
Embeds the Ruby interpreter into the Apache Web server
Include MySQL data into an Apache configuration file
Strong cryptography for Apache 1.3 via the SSL

phpWebFileManager
portmon
Robotcop
SkunkWeb
SwiftSurf
WWWOFFLE

Sound

AlsaPlayer
BeatForce
KAudioCreator
LyricDisplaySystem
MinidiscTocEditor
Mokion
mp3dup
MuSE
rioutil
SpiralSynthModular
Umix
XMMS
XMMSShell
YAMDAGER

System

cdbbackup
CUPS
Current
devfsd
DumpRestore
EVMS
FreeType
Ghostscript
gimp-print
grubconfig
LILO
modutils
nmbscan
OMNIDriver
rpmLint
Samba
UntzUntzLANScan
xhdbench

Simple file management tool, written in PHP
Monitor network services and notify when a server stops
Prevent spiders from accessing parts of your sites
Scalable, extensible, easy-to-use web application server
HTTP proxy that can spy, filter and modify HTTP requests
Web proxy for dialup Internet users

PCM player written with the ALSA sound system in mind
Computer DJing system, with two players
Audio file creation solution for KDE
Edit/display song lyrics on a second screen/projector
QT editor for the minidisc table of contents
Music player with advanced jukebox
Recursively looks for duplicate MP3 files
Mixing, encoding and network streaming of sound
Interface with Diamond Rios and the Nike psa[play
Object-oriented modular softsynth/sequencer/sampler
Adjust soundcard volumes and other mixer features
Multimedia player based on the look of WinAmp
Command line interface for controlling XMMS
Yet Another MiniDisc Manager

Easily make backups to CD-R or CD-RW media
Portable printing layer for Linux
Server implementation for Red Hat's up2date tools
Management of device entries in the device filesystem
System backup and restore commands
The Enterprise Volume Management System
High-quality and portable font engine
Processor for PostScript and PDF files
Collection of very high quality printer drivers
Script to help you install the GRUB bootloader
Boot loader for Linux and other operating systems
Utilities to make a Linux modular kernel manageable
Scans the shares of an SMB network
Support for over 400 printers using Ghostscript
Check for common errors on RPM packages
Serve files and printers to Windows clients
Creates a listing of files on your local Samba network
Tests the output speed of several devices

DVD

Neil Bothwick is your guide through the wonders of this month's jam-packed Linux Format DVD. The distros and dev tools should keep you busy, but don't forget the games and desktop toys.



Important notice

Before you even put the DVD in your drive, please make sure you read, understand and agree to the following: The Linux Format DVD is thoroughly tested for all known viruses, and is independently certified virus-free before duplication. We recommend that you always run a reliable and up-to-date virus-checker on ANY new software. While every care is taken in the selection, testing and installation of DVD software, Future Publishing can accept no responsibility for disruption and/or loss to your data or your computer system which may occur while using this disc, the programs or the data on it. You are strongly advised to have up-to-date, verified backups of all important files. Please read individual licences for usage terms.



Wherever you see this logo it means there's related stuff on the DVD

As with the first CD this month, the DVD contains a bootable installer for Trustix Linux. The distribution from the second CD, Sorcerer, is in the Distros directory as an ISO file.

Follow the instructions on the panel on p103 to burn this to a CD-R or CDRW, before booting from it.

Desktop

KDE3 is coming along, with a second beta released recently, so it is a good opportunity to see what the finished product will look like. DVD writers are coming down in price. *CDrecord* is the *de facto* CD writing package, and we have a patch for this to write DVDs, *dvdrttools*. If you regularly exchange files with Windows systems, you'll have come up against the problems of filename extensions. *ChangeSuffix* can add, remove or alter the extension for a batch of files, such as changing .htm to .html and back.

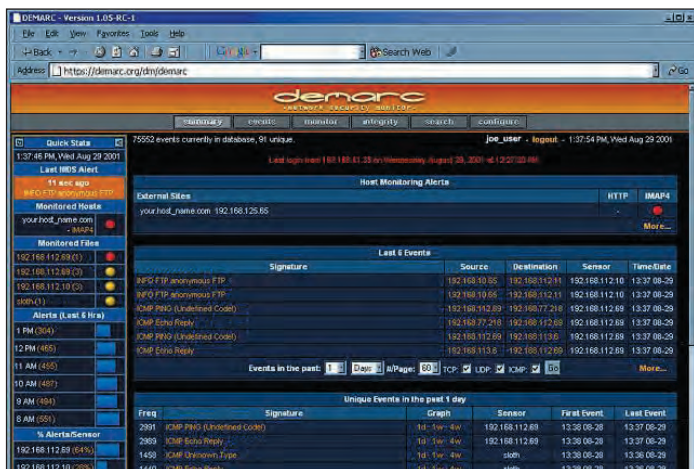
Development

The various database implementations all have small but significant differences in their SQL handling, which can make writing programs to work with them more work than it should be. *SimpleDBLibrary* and *liblookdb* both help with this by providing a common interface via a library. Whatever language you use, you'll need to be able to parse XML files at some time. This month Ruby users get *REXML*.

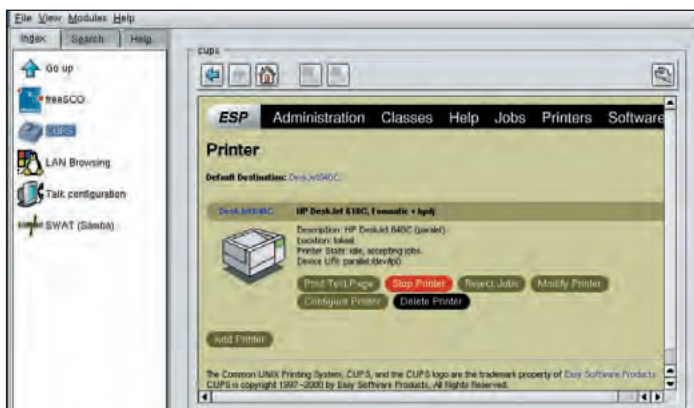
Internet

Usenet is not the ideal medium for distributing binary files, but that doesn't stop a lot of people using it for just that. Collecting and assembling

Defective CDs. In the unlikely event of your CD/DVD being physically damaged we'll send you a new, working version within 28 days. Send your defective disc – complete with your name, address, and a description of the fault – to: **LXF Disk Returns, TIB Plc, Unit 5, Triangle Business Park, Pentrebach, Merthyr Tydfil, Mid Glamorgan CF48 4YB.**



Dmrc network monitoring is in Magazine/Security, along with the other security programs.



Embed web-based configuration tools into the Control Centre.

multi-part uuencoded binaries is a tedious task, so we've got a program to do it for you, *brag*. Server languages, like PHP, make it easier to create large sites, but that's no help if your web host doesn't provide such a language. DML is a templating language to give similar help in this situation.

Server

The disadvantage of languages like PHP is that the server has to

repeatedly execute the script to serve essentially the same data each time a visitor requests a page. *AfterBurnerCache* is a caching system for PHP that will speed up serving of many PHP pages while reducing overall load on the server.

What do you do if you don't have access to your web host's *Apache* log files, or you can't run analysis tools on them? *PhpOpenTracker* logs page access statistics in a *MySQL* database, which means you can extract whatever information you want, when you want.

There's more

This is only a brief overview of this month's LXF DVD. Load up the DVD's index.html file to find the rest.

» DVD CONTENTS AT A GLANCE

Everything that is on the CDs, plus:

Desktop

bfr	Maintains a buffer of data collected from stdin
ccrypt	Encrypts and decrypts files and streams
ChangeSuffix	Changes the suffix of file and/or directory names
drsync	Perl wrapper for rsync
dvdrttools	Fork of cdrtools with support for writing to DVDs
e3	Full-screen, user-friendly text editor
Euler	Interactive computing with real and complex numbers
gEuCo	Converts currencies to and from Euros
glastree	Builds live backup trees with branches for each day
KDE3beta2	The second, and probably final, beta of KDE3
LogMon	Monitor several logs from one terminal window
PBBUTTONS	Evaluates the hotkeys on Apple iBook/PowerBook/TiBook
PocketPC	
ContactsImport	Import WinCE or PocketPC contacts into KDE
Rotix	Generate rotational obfuscations, like ROT-13
RubyDataqueryShell	Text-based shell for querying SQL databases
SciTE	Single-document editor using Scintilla
SharedExpenses	Web-based expense manager for a group of people
tpctl	Configuration tools for IBM ThinkPad laptop computers
ttyrec	tty recording and playback
WebConfig	Control Center module to embed a web browser
WebminLayout	A new and better navigation system for Webmin?
Xpdf	Viewer for Portable Document Format (PDF) files
xstroke	Full-screen gesture recognition program

Development

bonddb	Object orientated wrapper for PostgreSQL
CGI-Cache	Caches output of time-intensive CGI scripts
Evocosm	Extensible set of standard C++ templates and tools
GStreamer	Streaming media library and set of tools
JorisVoiceOverIPlib	Object-oriented Voice-over-IP (VoIP) library in C++
liblookdb	An interface to several Database Management Systems
libSIMD	Mathematical library using SIMD features of common CPUs
libspoc	Simple-to-use POP3 client library
NettleLibrary	Cryptographic library
Phpmole	Modularized development environment.
PHPMyEdit	PHP table preprocessor and interface generator
PonyProg	Serial device programmer with a user friendly GUI
REXML	XML parser written in and for Ruby
Scintilla	Source code editing component for Win32 and GTK+
SimpleDBLibrary	Support multiple database systems in an application
TclTk_ProjectManager	Dull IDE for programming in Tcl/Tk

Distros

Securepoint	Securepoint Firewall and VPN Server
SentryFirewall	Easy to maintain firewall or Intrusion Detection node
SMEserver	The gateway server formerly known as e-smith
Sorcerer	Source-based distro for advanced Linux administration
SuperRescueCD	Single very large bootable system-on-a-disk
Trustix	Boot from the DVD to install this

Games

AirTraffic	Air traffic control game/simulator
bantumi	Bantumi is a well-known game with an AI
FrozenBubble	Throw colorful bubbles and destroy them
MoxQuizz	Multilingual IRC quiz/trivia script
Q3SQLStat	Complete statistics suite for Quake3

Graphics

AxPoint	Generates PDF slideshows from a simple XML description
drawboard	Java applet used to make graphical teleconferences
Gimp	The GNU Image Manipulation Program
GIMPImagePlugin	Convert images from the Minolta DiMAGE digital camera
GreatCharts	Create stock charts, fpr web sites etc.
ImageMagick	Automated and interactive manipulation of images
JFreeChart	Java class library for producing charts
VideoDiskRecorder	Digital satellite receiver program using DVB

Internet

brag	Collects and assembles multipart usenet binaries
CheckUPS	Checks UPS' web site on the status of a package
DisSpam	Personal solution to combat spam

DML
EC
gaim-dict
Galeon
GNetFile
SharingSystem
Gwatch
Kppoe
Kshowmail
Kylie
mboxSecretary
Micq
Mozilla
nnNewsreader
Sylpheed
tinc
TuxCall

Office

AFT
Etoro
LocalSQL
mysqldiff
QuasarAccounting
XMLPublication

Server

afterBURNERCache
Anti-Web
Aphid
bind
citecast
colobus
CryptNETKeyserver
ERW
FTPmond
GCGI
Minimal-router
Moto
MySQL
oops
Papercut
PasTmon
PHPLayersMenu
phpOpenTracker
Postfix
PostgreSQL
ProMA
Pulsar
SASAccountingStats
Sitelite
TCLink
VorpaiMail
vqadmin
WeSQL
YAALA

Templating language for sites without PHP, Perl etc.
Email program using the Perl/Tk GUI libraries
Dictionary plugin for the Gaim instant messenger
GNOME Web browser based on the Mozilla rendering engine

Decentralized, distributed network
Mail spool monitor
KDE wrapper/configuration tool for PPPoE connections
KDE tool for watching for mail on POP3 servers
Kylie is a lightweight Web browser for X.
Generates statistics from an mbox-formatted mailbox
ICQ clone for the console
Web browser and rendering engine for other browsers
Curses-based USENET news reader
GTK+ based, lightweight, and fast email client
Virtual Private Network (VPN) daemon
Call waiting detector, hangs up the modem on a voice call

Almost Free Text is a document preparation system
Create readable, maintainable table definitions
Direct invocation and processing of SQL statements
Compares the data structures of two MySQL databases
Business accounting application
Generate Web pages from desktop documents

The first caching system for PHP4
Single-process Web server
Apache/Perl HTTP Installation Daemon
Internet name server
Content management system written in PHP
NNTP server for ezmlm mailing list archives
OpenPGP (RFC2440) compliant public key server
Handles complex databases using a Web browser
Email notification of uploads to an FTP server
NCSA's Common Gateway Interface
Linux-based router
Server-side scripting language like PHP or ColdFusion
Widely used and fast SQL database server
High performance HTTP-1.1/FTP proxy server
Multi-threaded NNTP server written in Python
Monitors your application servers
Hierarchical dynamic menu system
Site and visitor tracker using SQL storage
An alternative to the widely-used Sendmail program
Robust, next-generation, Object-Relational DBMS
PHP4-based system for administering a ProFTPd server
POP3 mail server with an intuitive configuration file
Fetches traffic stats via ipchains, ipfstat or iptables
Powerful web content manager and server
Client for running credit card transactions
Easily configurable but flexible sendmail replacement
Virtual Qmail Web Administrator
Use pure SQL queries directly in HTML files
Parses logfiles to generate detailed statistics in HTML

Sound

Gemu
gmmusic
KhdRecord
mhWaveEdit
OggEnc
RXSaturno
shorten
SloopSplitter

Configures sound cards based on the EMU10K1 platform
Database frontend to handle your entire music collection
Audio recorder that writes directly to hard disk
Edits sound files in Wave format
Command line encoder for the Ogg Vorbis format
Emulator for the popular Yamaha DX7 synthesizer keyboard
shorten is a lossy/lossless audio compressor.
Realtime sound effect program

System

Appcap
GENDIST
lwIP
pdumps
ProcessMonitor
PT
rpmcheck
Shadow
SmokePing
transitmount

Lets root read standard input and output of any program
Easily create your own special distribution
Lightweight implementation of a TCP/IP stack
Simple daily backup system
Kernel module to watch all programs executed
Easy-to-use, fast, and configurable printing tool
Check RPM packages and dependencies
Convert password files to the shadow password format
Network latency monitor
Simple rack/removable hard disk management

Help wanted

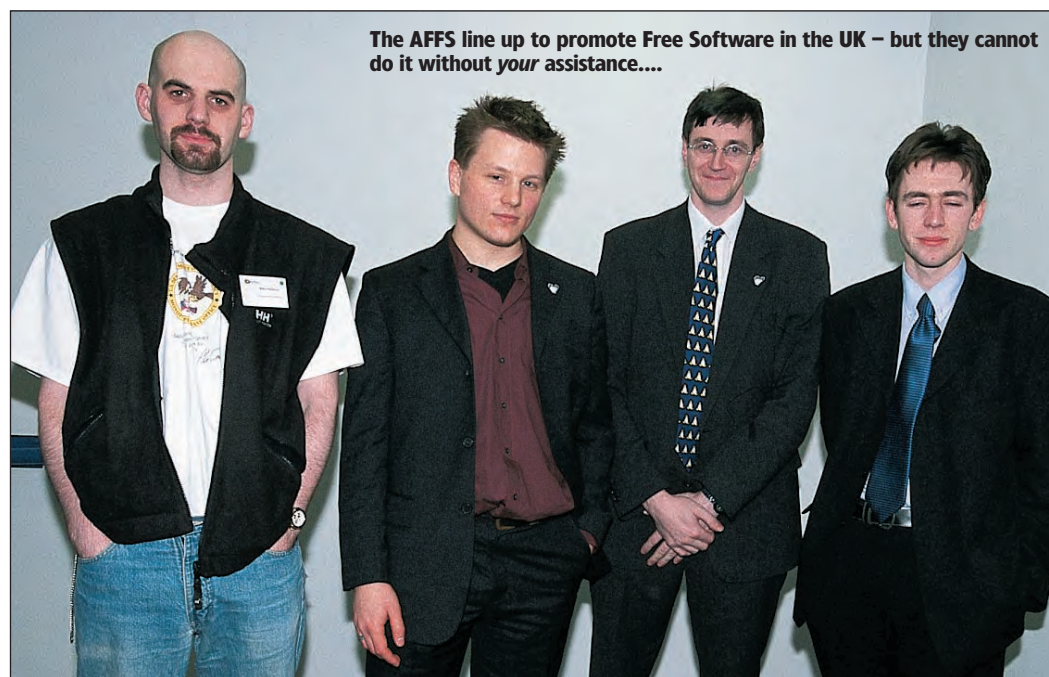
Free software doesn't just arrive shrink-wrapped and ready on the shelves of some out-of-town retail giant. It's a constant collaborative effort from a community that's happy to have you involved.

We start with a brief request to everyone in the UK. As you may have read (on *page 112*), the Sheffield Linux Seminar saw the formation of The Association For Free Software (AFFS) – an organisation seeking to support and promote Free Software within the UK. AFFS aim to collect a large membership of Free Software supporters to back them in representing the interests of the community at a national level.

As the organisation gears up for its official launch, and the launch of its first campaign, there are many tasks that need doing. In particular the threat of software patents.

By the time you read this the AFFS will be ready to accept membership – have a look at the website, to see what you can do to help Free Software in this country: www.affs.org.uk/ Or jump on board the mailing list and get involved: http://mail.gnu.org/mailman/listinfo/fsfe-uk

And while on the topic of advocacy, for anyone who missed the announcement from NCC (the UK's National Computing Centre), they are currently promoting Open Source in a series of regular media releases about their activities and position. If you have a view or experience that you would



The AFFS line up to promote Free Software in the UK – but they cannot do it without *your* assistance....

like included, then contact Michael Dean: Michael.Dean@ncc.co.uk

Sorcerer GNU Linux

If you've had enough time to try the impressive Sorcerer GNU Linux (SGL) from the coverdisc (reviewed on

page 31), you may thought about getting involved. SGL needs computers, money, and a suitable web host to distribute production quality sorcery from. More urgently, however, more than 100 spells are awaiting review and there is far too much work

for the project's founder – who has other commitments. If you can help then contact Kyle Sallee at cromwell@metalab.unc.edu <http://sorcerer.wox.org/> [irc://irc.openprojects.net/](http://irc.openprojects.net/) [#Sorcerer](#)

The Digital Divide

An ambitious new project

We have, in previous issues, highlighted how non-programmers can get involved in various projects, now here's one from a couple of non-programmers who desperately need the help of coders.

Tom and Steffi Russell are aiming to use Free Software to conquer the "digital divide." They aim to bring computing both to those without the resources for it and to those without the time to learn how to handle the complicated beasts with which LXF readers enjoy wrestling.

The project aims to work on slow Internet connections and old,

'recycled' hardware, yet still give a smooth and seamless computing and 'net experience to those not blessed with abundant resources.

The Vizor

At the heart of this Australian project is a dynamic interface called *The Vizor*, glueing together the (media) apps on the computer. It is a new interface paradigm – constantly displaying visual programs, during the time changes are loaded and processed for display, to give an uninterrupted media flow to the viewer, such as many non-computer users have come to expect

from other media, such as television.

The aims and design of *The Vizor* have been thought through in some detail – see the website. However the coding and, indeed, architecting of the application is awaiting the attentions of coders. This would make an extremely good project for a coding class, as well as for any highly skilled individuals who would like to get involved. There are few chances to get involved with a project at such an early stage and have a hand in the architecture – instead of having to rewrite other's buggy code. <http://www.geocities.com/myvizor/viz2/vcode/plan.htm>

Your help wanted

There are thousands of Free Software projects in need of some sort of help and thousands of *Linux Format* readers who may be interested in assisting your project. However we cannot publicise what we don't know about. If YOU have a project that's in need of anything from artists and beta-testers to web-designers and, er, something beginning with Z, we want to hear about it. Email us now at linuxformat@futurenet.co.uk and give us some details of your project, and what sort of help you are looking for. Please include plenty of info about the project!

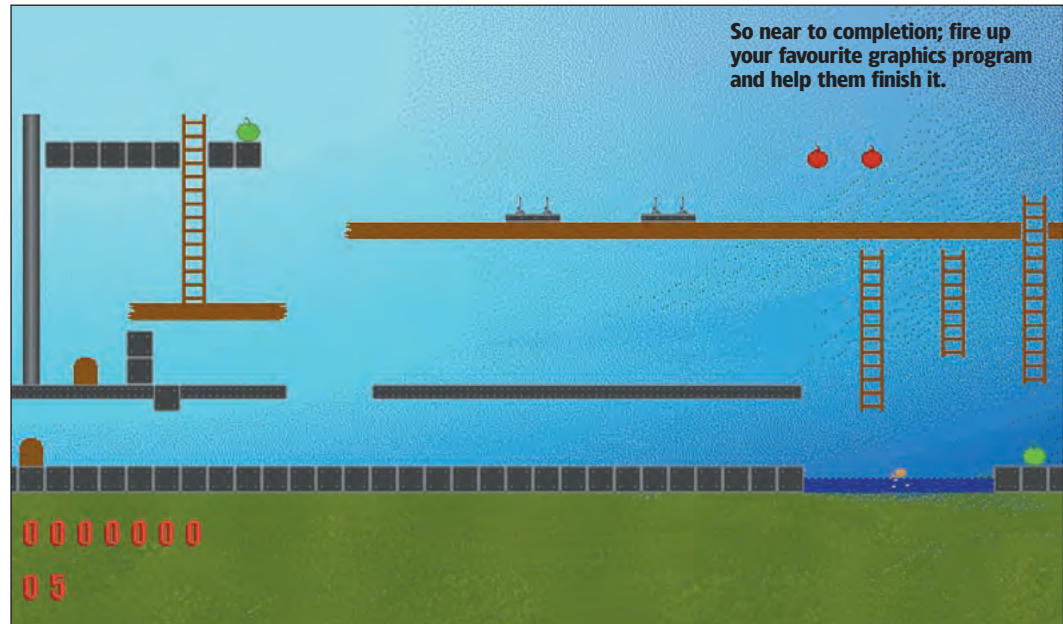
Tinyman

Steve Caddy writes on his near-complete platform game

Tinyman started life as a platform game written in Mallard BASIC for the Amstrad PCW, while I was still at school, some 6 or 7 years ago, by a guy called Peter Hurst. Its popularity among those that discovered it was quite staggering, and between Peter, another friend of mine, Tom McIntyre, and I, we decided a PC version would be a really good idea, since we would be able to put in lots of extra things that weren't practical on the PCW.

After a few false starts, we started writing the game in C as a DOS application using the MicroSoft C++ compiler. We ditched the MS compiler after we discovered that the Borland IDE was easier to use, and some of the Borland extensions were rather useful.

By the summer of 2000, with the advent of Windows 2000, creating 16 bit applications just wasn't seen as the done thing, so I proposed to port the existing (nearly complete) program to Linux, simply re-writing the IO modules. This caused a bit of a flag day, as the C was very badly written, and the header files were in a very bad way, with code all over the place. So Tom and I (since we were doing most of the developing anyway) restructured the code, and wrote the required modules to make the IO work under RedHat Linux, interfacing the



existing routines with *SVGALib* (making the game console based, although it should be quite easy to adapt it to run under X). We fixed a few other bugs along the way, and left the game in still a very nearly complete state.

In February 2001, I placed the code in CVS under SourceForge, so that we could share the code, and GPL'd what we had written (although

the headers of the files may still need changing to reflect this).

Since then, we've done almost nothing. The entire project has slowed to a grinding halt, because I was in the final year of my degree, and Tom was just starting his degree. There's not much that needs doing – a few levels need creating (there is a basic level editor, but it's not very user friendly –

I hacked it together in a few hours), and the odd bit of graphics... maybe a main menu, or something. The game engine has one or two known bugs, but other than that, it's complete.

So if anyone wants to help out to create graphics, convert it to run under X, or just to design some levels, just drop me an email.

dyn@m0ng.com

PicoGUI

A pocket sized display protocol

Micah Dowty writes:

"*PicoGUI* is an Open Source project to create a new GUI architecture for embedded systems. It provides the same functionality that you would expect from an X server, a widget set, and a window manager. Instead of compatibility with existing software, *PicoGUI* is focusing on an innovative architecture especially well suited to handheld computers, PDAs, cellphones, and other embedded systems. It is client-server, but almost everything is implemented in the server including widgets and themes. The theme system and flexible video architecture allow it to

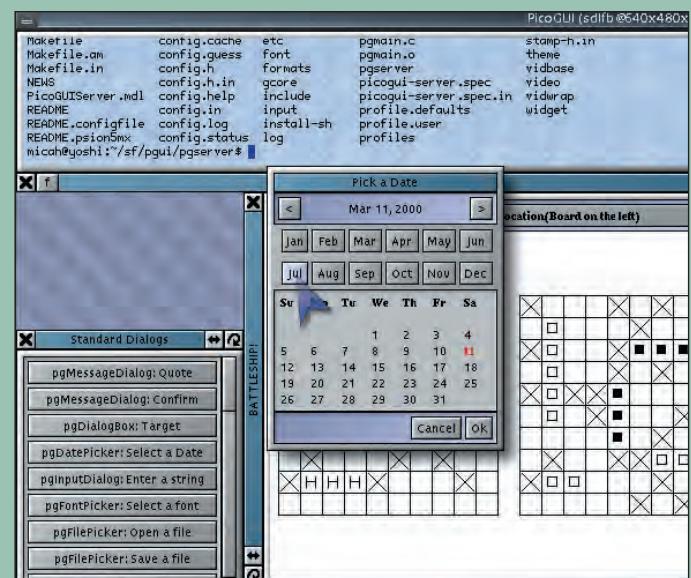


PicoGUI runs on all things small and neat.

run on a wide variety of displays, from large desktop monitors to text-only LCDs and everything in between.

"The project has been underway for almost two years – it runs on several architectures including the TuxScreen, VTech Helio, Agenda VR3, Psion, uCsim, and a few commercial products in development. However, there is still much to be done. *PicoGUI* needs more applications, mainly a PIM. We could use programmers to write applications, port *PicoGUI* to new architectures or programming languages, and work on *PicoGUI* itself. We're also in need of documentation authors, and artists."

<http://picogui.org>



Note the general alpha blended gradient goodness of the 3D theme.

User Groups

Your local Linux User Group needs you! LUGs worldwide are full of members keen to help with your problems, discuss ideas and generally natter about all things Linux. We have collected a load of information here so you can find the LUG closest to you. You can find lots more information online at: www.lug.org.uk or at www.linuxformat.co.uk/links.php

1 Hampshire

URL www.hants.lug.org.uk
Contact Hugo Mills

2 Bristol & Bath

URL www.bristol.lug.org.uk

3 Scottish

URL www.scottish.lug.org.uk
Contact Tony Dyer

4 Oxford

URL www.oxford.lug.org.uk
Contact Alasdair G Kergon

5 Bromcom (Kent)

URL www.kent.lug.org.uk
Contact John Mills

6 Brighton

URL www.brighton.lug.org.uk
Contact Johnathan Swan

7 Sussex

URL www.sussex.lug.org.uk
Contact Mike Pedley

8 Northants

URL www.northants.lug.org.uk
Contact Kevin Taylor

9 Anglian

URL www.anglian.lug.org.uk
Contact Martyn Drake

10 Milton Keynes

URL www.mk.lug.org.uk
Contact Denny De La Haye

11 Doncaster

URL www.doncasterlug.org.uk
Contact Andy Smith

12 Moray

URL www.moray.lug.org.uk
Contact Stewart Watson

13 West Wales

URL www.westwales.lug.org.uk
Contact Dan Field

14 Wolves

URL www.wolves.lug.org.uk
Contact Jono Bacon

15 Peterborough

URL www.peterboro.lug.org.uk
Contact Steve Gallagher

16 Edinburgh

URL www.edinburgh.lug.org.uk
Contact Alistair Murray

17 Tyneside

URL www.tyneside.lug.org.uk
Contact Brian Ronald

18 Leicester

URL www.leicester.lug.org.uk
Contact Clive Jones

19 Greater London

URL <http://glug.linux.co.uk/>

20 Surrey

URL www.surrey.lug.org.uk
Contact Jay Bennie

21 Cambridge

URL www.cam-lug.org

22 Devon & Cornwall

URL www.dclug.org.uk/
Contact Simon Waters

23 Falkirk

URL www.falkirk.lug.org.uk

24 Manchester

URL www.manlug.mcc.ac.uk
Contact John Heaton, Owen Le Blanc

25 Hertfordshire

URL www.herts.lug.org.uk
Contact Nicolas Pike

26 West Yorkshire

URL www.wylug.lug.org.uk
Contact Jim Jackson

27 Sheffield

URL www.shefflug.co.uk
Contact Richard Ibbotson

28 Staffordshire

URL <http://www.staffslug.org.uk/>

29 North East

URL www.shofar.uklinux.net/NELUG

30 London

URL www.lonix.org.uk

31 Thames Valley

URL www.sclug.org.uk

32 Liverpool OpenSource

URL http://linux.liv.ac.uk/_liv_linux_ug/
Contact Simon Hood

33 Deal Amiga Club

Email superhighwayman@hotmail.com
Contact John Worthington

34 Chesterfield

Email spirelug@yahoo.co.uk
Contact Robin Needham

35 South Derbyshire

URL www.sderby.lug.org.uk/
Contact Dominic Knight

36 Belfast (BLUG)

URL www.belfastlinux.cx
Contact Ken Guest

37 Wiltshire

URL www.wiltshire.lug.org.uk
Contact Jason Rudgard

38 South London

URL www.sl.lug.org
Contact Ben@benguin.co.uk

39 Cheshire

URL www.sc.lug.org.uk
Contact Anthony Prime — enquiry@sc.lug.org.uk

40 North Wales

URL www.northwales.lug.org.uk
Contact Jonathan Cole

41 Midlands

URL www.midlandsLUG.cjb.net WARNING: Popup ads
Contact Pete Thompson

42 Cumbria

URL www.cumbria.lug.org.uk
Contact Jamie Dainton

43 Dorset

URL www.dorset.lug.org.uk
Contact Beanz and Tracy

44 Shropshire

URL www.shropshire.lug.org.uk
Email shropshire@lug.org.uk

45 South West

URL www.southwestlug.uklinux.net/
Email southwest@lug.org.uk

46 South Wales

URL www.sw.lug.org.uk
Contact Tim Bonnell

47 North London

URL <http://www.kemputing.net/alt/lug/anlug.html>

48 Malvern

URL www.malvern.lug.org.uk
Contact Greg Wright

49 Huddersfield

URL www.hud.lug.org.uk
Contact Adam Brookes

50 Nottingham

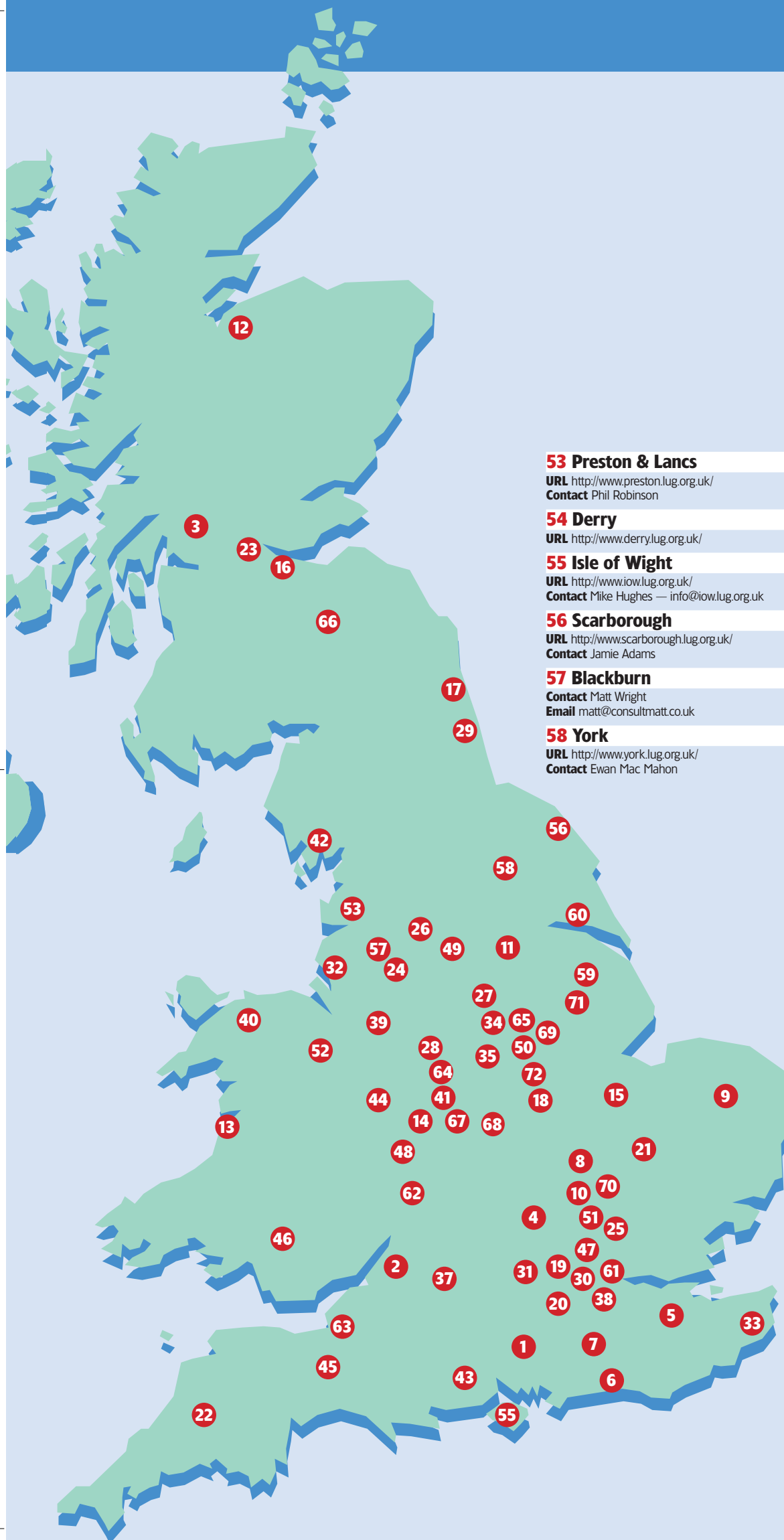
URL www.nottingham.lug.org.uk
Contact Godfrey Nix

51 St Albans & Luton

URL <http://www.lust.lug.org.uk/>
Contact Michael Culverhouse — mike@easily.co.uk

52 Wrexham

Contact Paul Kersey-Smith
Email paul@pkls.fsnet.co.uk

**53 Preston & Lancs**

URL <http://www.preston.lug.org.uk/>
Contact Phil Robinson

54 Derry

URL <http://www.derry.lug.org.uk/>

55 Isle of Wight

URL <http://www.iow.lug.org.uk/>
Contact Mike Hughes — info@iow.lug.org.uk

56 Scarborough

URL <http://www.scarborough.lug.org.uk/>
Contact Jamie Adams

57 Blackburn

Contact Matt Wright
Email matt@consultmatt.co.uk

58 York

URL <http://www.york.lug.org.uk/>
Contact Ewan Mac Mahon

59 Lincs

URL <http://www.lincs.lug.org.uk/>
Contact Chris Lingard

60 Hull

URL <http://www.hull.lug.org.uk/>
Contact Chrus Burton chris@hull.lug.org.uk

61 East London

URL <http://www.eastlondon.lug.org.uk/>
Contact Jonathan Spriggs

62 Gloucestershire & Cotswolds

URL <http://www.gloucs.lug.org.uk/>
Contact Barrie Haycock

63 Yeovil College

URL <http://www.yclug.lug.org.uk/>
Contact Adam Parker

64 South Staffordshire

URL <http://www.staffs.lug.org.uk/>
Contact Oliver Keenan

65 Mansfield

URL <http://www.mansfield.lug.org.uk/>
Contact Brent Vardy

66 Borders

URL <http://www.linux.bordnet.co.uk/>
Contact Welby McRoberts

67 South Birmingham

URL <http://www.sb.lug.org.uk/>
Contact Tim Williams

68 Coventry

Contact Darren Austin
Email info@coventry.lug.org.uk

69 Newark

URL <http://www.newlinc.lug.org.uk/>

70 Bedfordshire

URL <http://www.beds.lug.org.uk/>
Contact Neil Darlow

**NEW
DETAILS****71 Lincoln**

URL <http://www.lincoln.lug.org.uk/>
Contact Jon Shamash

**NEW
DETAILS****72 Loughborough**

URL <http://www.loughborough.lug.org.uk/>
Contact Martin Hamilton

**NEW
DETAILS**

LinuxUserGroups

LUG OF THE MONTH!

Shropshire

Shropshire LUG has a growing membership of around 45 at present. Backgrounds, ability levels and interests are quite diverse within the group, with members coming from a wide geographical area covering all of Shropshire, and over the borders into Staffs, Hereford and Powys too.

We holds monthly meetings for discussion, demonstrations

and talks as well as for co-ordinating the various projects underway within the LUG.

The group also run an IRC channel on undernet – #SLUG. Regular 'virtual meetings' supplement the main monthly meetings with real-time support and communication.

The group is currently investigating a range of projects, including building a

Wide Area Wireless Network between group members using inexpensive and home constructed equipment utilising Linux as the OS platform.

We are also extremely keen to work on other projects within the local and wider community, and always on the lookout for linkups with other user groups.

www.shropshire.lug.org.uk



Worldwide Linux User Groups

Free software users across the globe

Africa

PRETORIA

URL www.plug.za.org

Email andriesn@icon.co.za

STELLENBOSCH

URL www.entropysun.ac.za/

Email ixion@entropysun.ac.za

Australia

ADELAIDE LUG

URL www.linuxsa.org.au

Email mtippet@anu.edu.au

MELBOURNE, VICTORIA

URL www.luv.asn.au

Contact luv-committee@luv.asn.au

PERTH

URL plug.linux.org.au

Europe

AUVERGNE

URL www.linux-arverne.org/

Email Cyril.Hansen@wanadoo.fr

EIRE

URL www.linux.ie

Email root@linux.ie

URL www.dilu.org

Email glossary@dilu.org

GHENT

URL lsgg.rug.ac.be/

Email wvdputte@lsgg.rug.ac.be

GOTHENBURG

URL

nain.oso.chalmers.se/LUGG/index.html

LISBON

URL

www.students.iscte.pt/~a12593/gul.html

Email Paulo.Trezentos@iscte.pt

India

URL www.river-valley.com/tux/index.html/

Email anil@river-valley.com

URL www.linux-india.org

Email newsmaster@linux-india.org

North America

ALASKA

URL www.aklug.org/index.html

Email deem@wdm.com

BATON ROUGE

URL www.brlug.net/

Email dpuyear@usa.net

BAY AREA

URL www.balug.org/

Email aftyde@balug.org

CLARKSVILLE, TN

URL <http://www.clug.org>

Email tux@clug.org

DENVER

URL spot.elfwerks.com/~clue/

Email: lynnd@ihs.com

FLORIDA

URL www.flux.orgm

LOS ANGELES

URL www.lalugs.org/

Email dank@alumni.caltech.edu

NORTH COLORADO

Email nclug@nclug.org

Contact Mat Taggart

TAMPA

URL terrym.com/slug/index.html

Email paulf@quillandmouse.com

UHACC Normal, IL

URL <http://www.uhacc.org/>

Email lug@uhacc.org

VIRGINIA TECH

URL corvette.me.vt.edu/pages/index.html

Email nega@vt.edu

South America

BUENOS AIRES

Email dcoletti@impost.com.ar

LIMA

URL linux.unired.net.pe/

Email linux@unired.net.pe

MONTEVIDEO

URL www.linux.org.uy/

PARAGUAY/ ASUNCION

Email rolgiati@conexion.com.py

SAO PAULO

URL gul.linux.ime.usp.br/

Email gul@ime.usp.br

UK

Don't forget the distribution-specific mailing lists:

URL <http://www.lug.org.uk/maillist.html>

The Sheffield Linux Seminar



The ShefLUG room at the seminar provided geek refuge from the suits.

LUG events

Sheffield LUG organised the Linux Seminar to explain to business users the merits of GNU Linux as an OS and of Free Software as a better way of supplying and supporting businesses.

Organised with Business Link and Club UK Online, part of the DTI-sponsored UK online for business, who kicked off the day with a talk on the ICT directory, a demonstration of which showed a number of matches for "Sheffield" and "Linux suppliers".

SuSE's Roger Whittaker explained the history of Linux for those new to the OS. The highlight of the morning was the talk by FSFEurope President, Georg Greve, explaining why the freedoms inherent in Free Software are extremely

important for business users.

The afternoon saw an enthusiastic Julian Old, of Leeds Metropolitan University, giving a tco (total cost of ownership) case for GNU/Linux. Then Jeremy Allison spoke persuasively about "Linux's stealth weapon," *Samba*.

A very useful day, which certainly advanced the cause of Free Software with the local business community. Computer support for the day was provided by UKLinux.Net

<http://www.sheflug.org.uk/>

<http://www.clubukonline.co.uk/>

The meeting was also notable for the formal founding of the Association For Free Software (AFFS), who are now busying themselves for their official launch. www.affs.org.uk

Linux User Group organisers

If you're not listed here, or we have your details wrong, please contact us at: **LUGS!, Linux Format, 30 Monmouth Street, Bath, BA1 2BW** or email your details to: linuxformat@futurenet.co.uk

EDITORIAL**Editor** Nick Veitch nick.veitch@futurenet.co.uk**Art Editor** Julian Jefferson julian.jefferson@futurenet.co.uk**Reviews Editor** Richard Drummond richard.drummond@futurenet.co.uk**Production Editor** Richard Smedley richard.smedley@futurenet.co.uk**Editorial Contributors**

Jono Bacon, Neil Bothwick, Chris Brown, Steve Caddy, David Cartwright, Andy Channele, David Coulson, Dan DeNicolò, Micah Dowty, Hoyt Duff, Simon Goodwin, Maurice Kelly, Jon Kent, Brian Long, Biagio Lucini, Neil Luccock, Charlie Stross

COVER CD PRODUCTION**CD Editor** Neil Bothwick**Senior CD Editor** Kate Hadley**ART CONTRIBUTORS****Photography** Rick Buettner, Kath Lane-Sims, Photonica, Powerstock, SPL**Illustration** Paul Bateman, Chris Winn, Shane Collinge**ADVERTISING SALES**

Matt Stanley 01225 442244

matt.stanley@futurenet.co.uk

Emma Mimmack 01225 442244

emma.mimmack@futurenet.co.uk**MARKETING AND PROMOTIONS****Marketing Manager** Fiona Tully**Subscriptions Manager** Nick Skardon**PRODUCTION****Production Co-ordinator** Diane Ross**Group Production Manager** Clare Tovey**Advertising Production Co-ordinator** Jo Crosby**MANAGEMENT****Publisher** Dave Taylor**Group Advertising Manager** Simon Moss**Group Publisher** John Weir**Circulation Director** Sue Hartley**Managing Director** Colin Morrison**DISTRIBUTION AND CIRCULATION****Circulation Manager** Jamie Malleyjamie.malley@futurenet.co.uk**Distributed by** Seymour Distribution, 86 Newman Street,

London W1T 3EX

Tel +44 (0)207 3968000**Overseas Distribution** by Future Publishing Ltd.**Tel** 01225 442244.**Overseas Licences**Simon Wear simon.wear@futurenet.co.uk**Tel** +44(0)1225 442244 **Fax** +44 (0)1225 732361**Contact Details**

Linux Format, 30 Monmouth Street, Bath BA1 2BW

Tel +44 (0)1225 442244 **Email** linuxformat@futurenet.co.uk**Subscriptions and Mail Order****Phone** +44 (0)1458 271178. See page 100**Email** linuxformat.subs@futurenet.co.uk

Copyright No part of this publication may be reproduced without written permission from our publisher. We assume all letters sent – by email, fax or post – are for publication unless otherwise stated, and reserve the right to edit contributions. All contributions to Linux Format are submitted and accepted on the basis of

non-exclusive worldwide licence to publish or license others to do so unless otherwise agreed in advance in writing recognises all copyrights and trademarks. Where possible, we have acknowledged the copyright holder. Contact us if we haven't credited your copyright and we will always correct any oversight.

All CD-ROM demos and reader submissions are supplied to us on the assumption they can be incorporated into a future covermounted CD-ROM, unless expressly stated to the contrary. We cannot be held responsible for mistakes or misprints. Linux Format recognises all copyrights in this issue. Where possible we have acknowledged the copyright holder. Please contact us if we have failed to credit copyright.

Disclaimer All tips in this magazine are used at your own risk. We accept no liability for any loss of data or damage to your computer, peripherals or software through the use of any tips or advice.

Printed in the UK by Midway Clark (Holt) and © Future Publishing Ltd 2002

LINUX is a trademark of Linus Torvalds, GNU/Linux is abbreviated to Linux throughout for brevity.

All other trademarks are the property of their respective owners

Future Publishing Ltd. is part of The Future Network plc.

The Future Network produces carefully targeted specialist magazines and websites for groups of people who share a passion. We aim to satisfy their passion by creating titles that offer superb value for money, trustworthy information, multiple ways to save time and money, and are a pleasure to read or visit. Today we publish more than 80 magazines and over 30 magazine websites and networks from offices in five countries. The company also licenses 32 of its titles resulting in over 60 local editions in a further 23 countries. The Future Network plc is a public company quoted on the London Stock Exchange (symbol: FNET).

Non-executive Chairman: Roger Parry**Chief Executive:** Greg Ingham**Chief Operating Officer & Managing Director, UK:** Colin Morrison**Group Finance Director:** John Bowman**Tel** +44 (0)1225 442244**www.thefuturenetwork.plc.uk**BATH • LONDON • MILAN •
NEW YORK • PARIS •
SAN FRANCISCO • WROCLAW

NEXT MONTH

Issue 27 on sale Friday 26 April



COOLTOWN

HP's strategy for pervasive web technology envisages a world of talking bus stops and doors which know who you are – but how far away is this futuristic fantasy, and what part does Linux have to play?

THE FUTURE STARTS HERE, NEXT MONTH.

» Award winners

Who was the greatest, the best and most popular in our LXF awards? We welcome you to next issue's grand event, where the winners grab their virtual gongs and the losers put on a brave face.

LINUX
format
awards
2001

PLUS:

Reviews including *Heavy Metal:FAKK2*, *Money Dance*, *ESP Print Pro*, an expanded help section and more tutorials on everything from *Apache* to *VPN*. And don't forget, we'll have discs full of Linux goodness for everyone.

DON'T MISS YOUR COPY OF LINUX FORMAT!

Get it delivered to your door every month – subscribe on page 100