PROUDLY INDEPENDENT SINCE 2013

LINUXVOICE

October 2016 FREE SOFTWARE | FREE SPEECH www.linuxvoice.com

Banish Twitter!

Build your own social network to share daily blatherings

elgg

Why you need to try...

fedora

- The best performance
- The newest technologies
- The most passionate community

BOINC: HELP SCIENTISTS DO SCIENCE BETTER P72

GAMES EMULATION

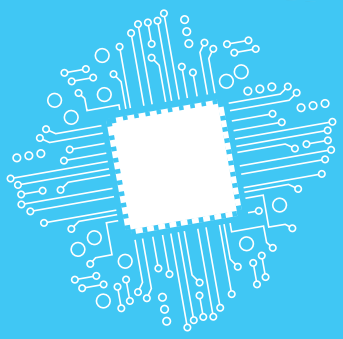
RetroPie

Enjoy the glory days of gaming all over again – on a humble Raspberry Pi

FRUGALWARE

Hardware

Put old hardware to good use with Linux and Free Software



FOSSPICKS – THE BEST FREE SOFTWARE FOR LINUX P64

BURSTING WITH AWESOME TUTORIALS!

SOLR Build a searchable archives of your chaotic PDFs
PHOTOGRAPHY Advanced tricks for image processing
VERACRYPT Keep your top-secret cat videos safe from The Man

MARTIN WIMPRESS

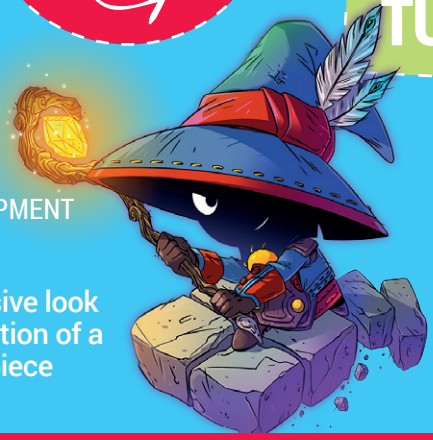
MR MATE

Why we have this man's family to thank for Ubuntu Mate

GAMES DEVELOPMENT

LUMO

Take an exclusive look inside the creation of a minor masterpiece

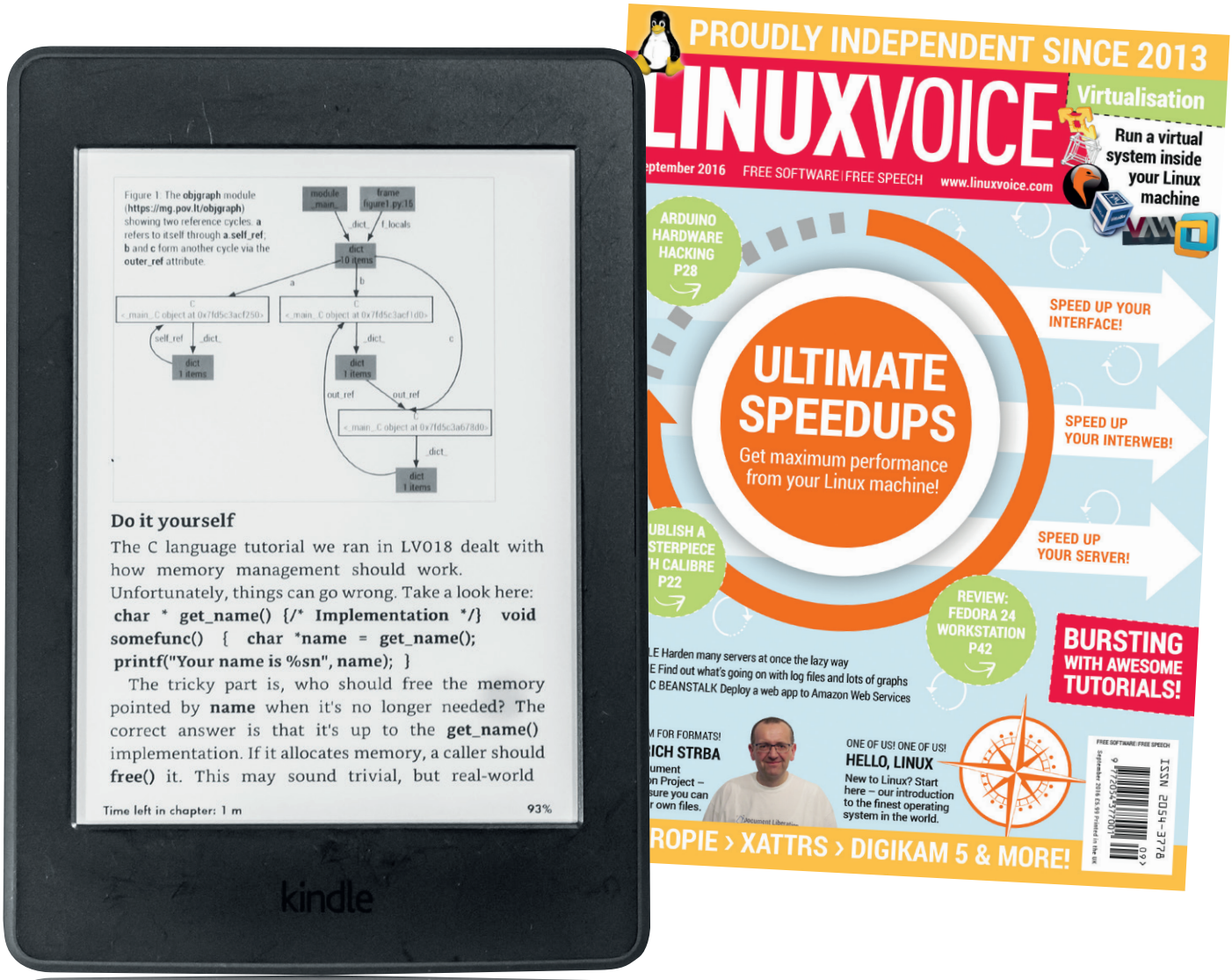


STELLARIUM › AFTERSHOT › OPENVR & MORE!

FREE SOFTWARE | FREE SPEECH
October 2016 £5.99 Printed in the UK
ISSN 2054-3778

LINUXVOICE

The only Linux magazine available as DRM-free PDFs and ePub



shop.linuxvoice.com

FREE SOFTWARE AHEAD

The October issue

BEN EVERARD

Long-term Linux user and best-selling author Ben is usually found knee-deep in either Python code or a tangle of wires.

This month I've been thinking a lot about what makes a distribution great, as I've taken a close up look at Fedora for the cover feature and interviewed Ubuntu Mate's Martin Wimpress. I'd love to say that I've come up with some deep insight that we can use to make all distributions perfect, but I haven't. There are lots of trade-offs to make...

Some distros do a great job of packaging lots of latest upstream software. Others do a great job of curating software so only the best in class in each area is available. Some are constantly changing, while others guarantee a stable base for a long period of time. Rather than try to work out what is objectively the best distro, you should think about what's important to you so you can find a distro that shares your values.

Ben Everard
Editor, Linux Voice

THE LINUX VOICE TEAM

Editor Ben Everard
ben@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Editor in hiding Graham Morrison
graham@linuxvoice.com

Editor at large Mike Saunders
mike@linuxvoice.com

Games editor Michel Loubet-Jambert
michel@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Mark Crutch, Simon Phipps, Les Pounder, Mayank Sharma, Nate Drake, Alexander Tolstoy and Valentine Sinitsyn

Linux Voice is different.
Linux Voice is special.
Here's why...

1. At the end of each financial year we'll give 50% of our profits to a selection of organisations that support free software, decided by a vote among our readers (that's you).

2. No later than nine months after first publication, we will relicense all of our content under the Creative Commons CC-BY-SA licence, so that old content can still be useful, and can live on even after the magazine has come off the shelves

3. We're a small company, so we don't have a board of directors or a bunch of shareholders in the City of London to keep happy. The only people that matter to us are the readers.

What's hot in LV#031

ANDREW GREGORY

I'm pretty sure that computer games reached their peak on the SNES. I'm going to give up on new games and take Mike's advice on retro gaming with the Raspberry Pi and RetroPie.
p22

GRAHAM MORRISON

I've never liked Facebook, but I do want to keep in touch with people. I've decided to run my own social network, so I've been reading Mayank's group test to find the right platform.
p56

MIKE SAUNDERS

Networks have always been a bit of a mystery to me. Usually everything just works, but when it doesn't, I'm stuck. Valentine's Core Tech this month has helped me understand what's going on.
p94

Contents

Make your computer great again!

Regulars

Cover Feature

Server, cloud or desktop – Fedora can do it all. Discover the cutting-edge features that the other distros won't have until next year.

Interview

Martin Wimpress

The man behind Ubuntu Mate on being an unpaid family tech support person.

Feature

RetroPie

Easy-peasy retro games emulation for your Raspberry Pi, wrapped in a single distro install. Let's play!

FAQ

Group Test

SECRETS OF PiTVi
TURN TO PAGE 26

Feature

28

Diary of a developer

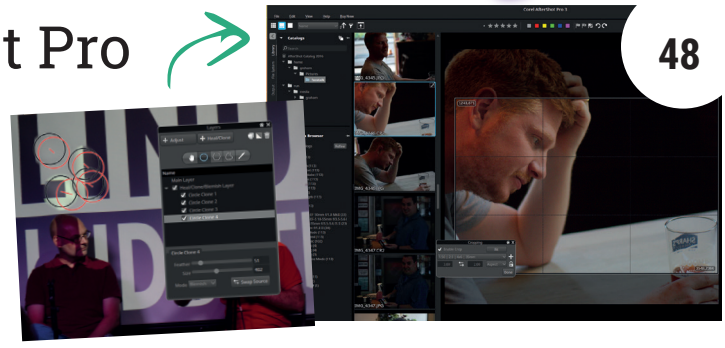
Inside the creation of this isometric puzzle fest.

Reviews

AfterShot Pro

48

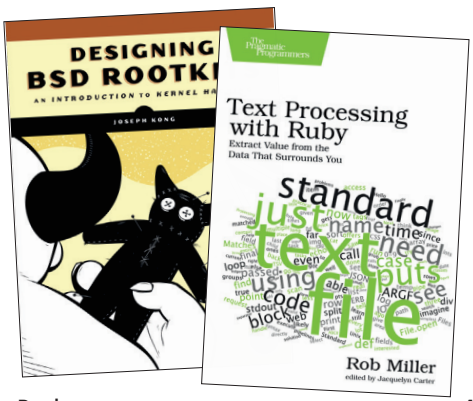
Take photographs, then make your images better than you could have imagined with this feature-packed RAW image editor.



LibreOffice 5.2 43
Yet again, the flagship office suite gains more features and more polish, this time with a focus on collaboration.

Stellarium 50
Look outside: if it's daytime, or cloudy, or you live in a built-up area, you uprobably can't see the stars. That's why *Stellarium* is here!

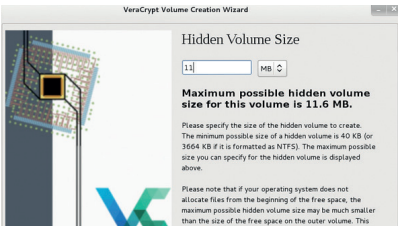
Gaming on Linux 46
Go back to the future with the 8-bit charm of *Undertale* plus a refreshed version of an all-time classic: *Day of The Tentacle*.



Books 48
Once the forests have gone, we'll need clever people to repair the broken environment. So read more – expand your mind while you still can!

Tutorials

Boinc 72
Contribute a few spare CPU cycles to help scientists solve climate change, synthesize proteins and more.

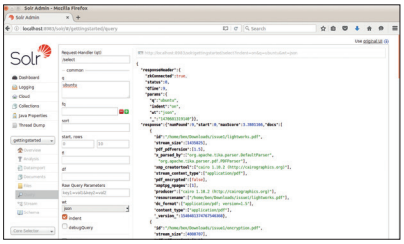


Veracrypt 74
Protect files, folders and partitions with serious encryption. Keep your cat videos secret, keep them safe…

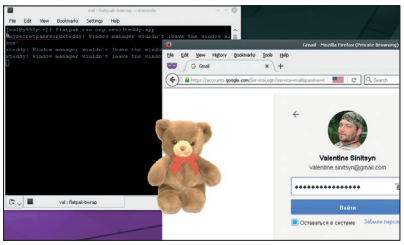
Raspberry Pi 78
Take your first steps into physical computing with a Raspberry Pi, a couple of LEDs and a smattering of Python.

Photography 82
Processing and editing tricks to improve your holiday snaps and fix those grainy old photos scanned from old film.

Coding



Solr 86
Build a search index for your PDF archive. Perfect for magazine back issues!



Packaging 90
Understand the magic that goes into packaging an app with Flatpak.

NEWS ANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

Public software

Developers don't need to mess about with legal stuff – we can do that for you.

Simon Phipps is ex-president of the Open Source Initiative and a board member of the Open Rights Group and of Open Source for America.

Back in the January issue, I talked about thse proliferation of Foundations for open source projects. All the most important freedoms – to use the software for any purpose, study and improve it and share with anyone – are secured by using an OSI-approved licence. But open source projects that develop beyond the first commits do end up needing a set of capabilities, including:

- An "Asset Lock", guaranteeing that community assets can only be used in ways the community approves (including domain names, trademarks and copyrights).
- A "bank", handling donations, paying staff and fulfilling tax-reporting obligations.
- An impartiality guarantor, anchoring the representation of its community and ensuring decisions are made in the way that the community wants independently of any one participant.
- An infrastructure provider, hosting code, mailing lists, forums and bug trackers and also hosting events.

Without a seperate legal entity to look after these things, there are plenty of examples of projects where a disagreement between the founders has led to a serious conflict that can have an unfair outcome based on who accidentally ended up as the holder of the assets. If that asset holder is the employer of one of the participants, things can get even more ugly.

Don't go it alone...

It's a natural instinct for makers to try to fix their own problems directly, so it's not surprising to find so many developers expecting the next step for their collection of co-developers is to start a non-profit association of some sort – a "foundation" – to host their project. But experience shows that making a new legal entity is often as poor a solution to the problem as remaining unincorporated. Running a not-for-profit entity takes time, experience and administrative attention. Moreover, once the activity scales, all of those are needed in amounts that distract from development.

For some projects, that's a price worth paying. The Document Foundation was created in just such circumstances, based on the expectation that many people would be willing to share the funding and running of a large, independent legal entity. But not every project is like that, least of all on day one. For most projects, the best choice is to move into the shelter of an "umbrella" Foundation, until the project grows to the point that an independent entity is feasible.

In the USA, there are a few of these "umbrella" entities that provide shelter for assets and governance. Software in the Public Interest was set up many years ago to host the Debian project's assets, and there's also the Software Freedom Conservancy, which provides a broader range of services, and now hosts a number of important projects. Both have a US focus, as well as being very much in demand.

That's why it seems a good idea to start an umbrella entity for Europe. We chose to start it in the UK where the Community Interest Company legal structure seemed perfect for open source projects. Thus Public Software CIC was created!

"Public Software" refers to software that members of the general public can use without restriction, improve as they wish and share with anyone. By using the name "Public Software" we are alluding to the name of the key free software and open source licence, the GNU General Public Licence (GPL) — a copyright licence for the benefit of the general public. Public Software CIC exists to promote software freedom delivered under any copyright licences, both approved by the Open Source Initiative as conforming to the Open Source Definition and recognised by the Free Software Foundation as "free software licences".

Public software means more than just the essential of an acceptable copyright licence. Our projects also allow participation by anyone and reserve project governance to those actually participating. We are convinced that code and community created together in public are the best recipe for software freedom. We expect some projects to move on to their own CIC or charity once they are established, but hope many will stay to form a critical mass of shared resources for the benefit of all.

It's early days for Public Software CIC but projects are already exploring membership. I hope this will become a force for good through software freedom in the UK and Europe. You'll find us at **https://PublicSoftware.eu**.

> It's natural for makers to try to fix their own problems, so it's not surprising so many developers... start a non-profit association

LibreOffice 5.3 • Skype • Bulgarians • Firefox encryption • Let's Encrypt •Lumina

CATCHUP Summarised: the biggest news stories from the last month

1 Debian sites now available as privacy-friendly onion services

Hidden onion services on the *Tor* anonymity network are often associated with dodgy shenanigans, but there are legitimate uses for them as well. Now the Debian GNU/Linux distribution has set up many of its websites as hidden services, so you can access them completely anonymously. The full list is available at **https://onion.debian.org** – including news, blogs and security updates. Anything that protects our online privacy is welcome.

2 Linux comes to the Microsoft Surface 3 tablet

Many of us in the GNU/Linux world aren't big fans of Microsoft software (to put it mildly), but some of the company's hardware isn't too shabby. If you've picked up or been given a Surface 3 tablet but want to move away from Windows, you may soon be in luck – Linux kernel 4.8 will add hardware support for the machine. The biggest change in that kernel release is support for the tablet's touchscreen, arguably the most important driver.

3 LibreOffice 5.2 released

Sporting new document classification features, OOXML signature import and a bunch of user interface improvements, *LibreOffice 5.2* is a decent upgrade and keeps moving way ahead of the mostly abandoned *Apache OpenOffice*. See our review on page 49 for the full lowdown on what's changed in this version.

LibreOffice
The Document Foundation

4 Bulgarian government goes pro open source

Another win for free software! The government of Bulgaria has enacted its Electronic Governance Act, which includes the following clause: "computer programs must meet the criteria for open source software". So anything written for use inside the government must be FOSS and accessible via a public repository. Not only could this save Bulgaria money in the long run, but it's an important step towards transparency and open standards for the country as well.

5 Skype for Linux returns – well, sort of

Ever since Microsoft bought Skype, questions have been raised about the software's long term future on Linux and other non-Windows platforms. Indeed, it looked like the Linux client was dead not so long ago. But now it's back: Microsoft has announced Skype for Linux Alpha, a "not fully functioning Skype client" that's available in Deb and RPM formats. Helpful to Skype users, then, but we much prefer open and standardised communication protocols.

6 Firefox 50 will trust Let's Encrypt by default

Let's Encrypt is awesome: it's a free SSL/TLS Certificate Authority (CA) which makes it possible for many websites to enable HTTPS (secure browsing) without having to shell out lots of money. Until now, Let's Encrypt has piggybacked on another CA, but wants to ultimately become completely independent. When *Firefox 50* is released in November, it will support Let's Encrypt by default – and hopefully other browsers will follow suit in time. Your move, Google, Microsoft, Apple...

7 Lumina Desktop 1.0 released

You could argue that we have enough desktop environments already, but Lumina is worth keeping an eye on nonetheless. It originally came to life in PC-BSD, a desktop-oriented flavour of the FreeBSD operating system. Now it's available on other Unix-like OSes as well, including various Linux distributions, and is relatively memory-friendly when compared to the mighty Gnome and KDE. For more information and screenshots galore, see the website: **https://lumina-desktop.org**.

8 Mozilla awards $585,000 to nine open source projects

The Mozilla Open Source Support Initiative has channelled funding into various FOSS projects that "meaningfully advance the Mozilla mission", giving them a development boost. Some of these are related to infrastructure that Mozilla uses; others are for security software. For the full details, see **http://tinyurl.com/z6t9rl5** – you can even put your own Free Software project forward for a funding request there too.

DISTROHOPPER

What's hot and happening in the world of Linux distros (and BSD!).

Clear Linux

Intel's container-based distro.

Intel has quietly been working away on a cloud computing distro for some time now with relatively minimal publicity in one of those rare cases where a big company develops its own Linux distro and actually retains the Linux name. The aim of the project is "not to provide yet another general-purpose Linux distribution", but rather a very fast container-based distro much like CoreOS, using what it calls "Clear Containers" while using "the isolation of virtual machine technology". The distro does not ship with a GUI by default, instead offering separate "bundles" offering a range of things, among them being a full Xfce desktop, enabling the user to pick and choose and keeping the distro light.

Clear Linux is a rolling release and updates constantly with over 20 builds per week. Recent updates include performance improvements, Mesa 3D 12.0 and kernel 4.6.4. Being a distro developed by Intel, it doesn't play nice with AMD hardware and doesn't package Radeon drivers. The flipside is that it is extremely well optimised, particularly in the realm of integrated graphics, providing significant performance boosts over conventional Linux distros. Clear Linux's "bundles" are based upon RPM, but use **swupd** built atop **bsdiff** and are conceptually different to conventional packages. There are a number of images available, including KVM-based images, live images and cloud deployment images.

Clear Linux does away with a lot of legacy baggage, which also boosts performance.

Installation is not that straightforward and is comparable to an Arch install in that you're greeted with a command line and the aforementioned bundles must be installed to add further functionality. ClearOS is meant as a showcase to demonstrate just what Intel's hardware is capable of under the right conditions, though it could be put to good use in a data centre.

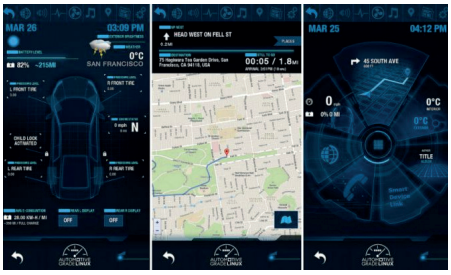
Automotive Grade Linux

Linux under the hood

OK, so you might not want to install this on your PC unless it has wheels and an engine, but Automotive Grade Linux (AGL) is a fascinating project supported by the Linux Foundation that aims to power the car of the future and its entertainment system.

The foundation has clearly recognised the potential and widespread use of open source operating systems and software in an era where the days of companies building their own software from scratch is mostly behind us. Instead, AGL takes an 80/20 approach, in which the vast majority of the system is provided, while the automotive manufacturer tweaks and adds the rest based on the features they want, also adding their own branding. Some of the partners include Toyota, Honda, Mazda, Ford and Nissan, as well as sound system suppliers like Panasonic.

Version 1.0 was showcased at the Consumer Electronics Show (CES) in January 2016 and made quite a splash, and the first cars running the distribution should be hitting the streets by 2018. This makes a lot of sense given how much cars have lagged behind, often using slow and clunky systems. Although Apple CarPlay and Android Auto already exist, AGL has some big advantages in being fully open source and having so many industry backers.



The current release drives the entertainment system; future releases will also drive the car.

News from the *BSD camps

What's going on in the world of FreeBSD, NetBSD and OpenBSD.

In the OpenBSD world, version 5.9 of LibertyBSD has been released. LibertyBSD aims to remove all non-free elements from OpenBSD in a "deblobbed" version of the OS – an approach not often seen in the BSD realm, which has traditionally differed from the puritanical approach of the Free Software Foundation. The biggest change in version 5.9 is that support for i386 has been added, where previously only amd64 was supported, while other changes include additional "deblobbing". On OpenBSD itself, **usermount** is being removed, thus removing the ability for non-root users to mount filesystems, citing security concerns as a result of a number of bugs relating to it, which have been described as "the tip of the iceberg".

On the FreeBSD side of things, work is going on to bring AMD's newer graphics binaries to the system and porting **amdgpu** from Linux 4.6. This is close to being completed and when finished will mean that FreeBSD users will have access to much newer AMD graphics hardware. Also in FreeBSD, it is now getting to the point where replacing OpenSSL with LibreSSL is becoming a viable option. This would be a big step for web developers looking to close security vulnerabilities; however, it is far from being standard on a FreeBSD base and patches are currently required. FreeBSD 11 has also entered beta at the time of writing and the final release is expected soon.

LibertyBSD takes a hard-line approach to non-free binary blobs comparable to that taken by the Free Software Foundation and distros such as gNewSense and Trisquel.

NAS4Free, a FreeBSD 10.3-based embedded storage distribution which aims to provide the famed BSD reliability and longevity in lieu of store-bought boxes, has seen a new release with a number of changes. These changes include the added ability to set and change UPS monitoring credentials, support for deleting multiple items at once in the snapshots page and adding a search function to Diagnostics|Log.

AtheOS – discontinued and forgotten

AtheOS is one of those forgotten dead ends that had a relatively short shelf life of four years from 1997 to 2001, when it was discontinued. It was originally designed to be a clone of AmigaOS to provide some continuity given the dire situation Commodore was in at the time and also making use of the GPL, releasing the system as free software.

Despite this initial aim, it was soon abandoned in favour of a general system that aimed to be largely POSIX compliant, but not a Unix clone. One major departure from Unix-like operating systems of the time was not using the X Window System, instead having its own integrated GUI built from scratch. This meant it was more responsive, at the cost of being able to easily port readily available applications. Other components, like the kernel, filesystem and web browser were also created from scratch.

Development had become stagnant on the project by 2002 when the main developer began to dedicate his time to other things, and another group of developers decided to create a fork of the OS known as Syllable Desktop. Syllable picked where the short-lived AtheOS left off, thanks to the GPL-published source, and is still in reasonably active development and installable on modern hardware, available at **http://web.syllable.org**.

AtheOS' *ABrowse* web browser reminding us what the internet was like in 2000.

YOUR LETTERS

Got an idea for the magazine? Or a great discovery? Email us: letters@linuxvoice.com

STAR LETTER

THANKS!

I know you don't make Linux, but I thought this was as good a place as any to publicly air my thanks for Linux existing, specifically Linux Mint.

I've been working with a new-ish laptop that I bought last year. Like most people I didn't pay too much heed to the operating system – the term itself was a new one to me until recently. What prompted me to learn what an operating system is was the number of intrusive messages that Microsoft kept sending me about upgrading my version of Windows. I'd put some effort into learning how it worked and I didn't want to have to put that effort in all over again learning a new system.

I mentioned this to a friend who suggested I give Linux Mint a try. I only ever use the machine for email and using the internet, and he assured me that Linux could do everything that Windows can, so I gave it a go. In contrast to Windows, I haven't had to learn anything new at all – things just work the way I expect them to do. Using Mint feels like the first time I picked up an iPod – nobody needs to explain it to you, because you can figure it out for yourself just by playing with it. That's truly amazing, and the fact that I didn't have to pay any money for it is the icing on the cake.

David McKenzie, Sheffield

Andrew says: Cheers David! Linux is indeed ace, and we can't take any credit for it at all. However, we know someone who can take a little bit of credit – Martin Wimpress, who we speak to on page 40. If you like Linux Mint, you might like Ubuntu Mate. It was born out of a desire to not reinvent the wheel, so to speak, so it sounds like just the sort of thing for a chap who has better things to do than re-learn where the Start button is.

AND NOW FOR SOMETHING COMPLETELY DIFFERENT

I know Python is popular with new users, but I'd really like to see some other languages in Linux Voice from time to time. How about explaining the LAMP stack, with *MySQL*, PHP, Perl and a bit of *Apache* configuration, for example? I don't think messing about with LEDs and GPIO pins will help my career too much...

Sean Dwyer, Dublin

Andrew says: You can please some of the people some of time, but try any more than that and you're asking for trouble. We did run a couple of Perl tutorials last year, but we ended up getting flak from Perl programmers for doing it one way when we should have been doing it another way – this, really, is why we stick with a more welcoming language (Python) for the entry-level stuff.

I love the idea of a LAMP series of tutorials; we have the knowledge, it's just a matter of puling it all together, so watch this space. And in the meantime, if it's PHP you want it would be remiss of us not to direct you to **www.hackingwithphp.com**, a brilliant series of tutorials written by our former comrade Paul Hudson. He'll take you from newb to ninja!

PHP is an old language, but it works.

SPACE!

The Apollo 11 guidance code is up on GitHub. Well done NASA for sharing with us; it's really made me appreciate how far I have to go to become a decent programmer, and how much modern languages do for you. Now back to playing *Elite Dangerous*...
James Newsome

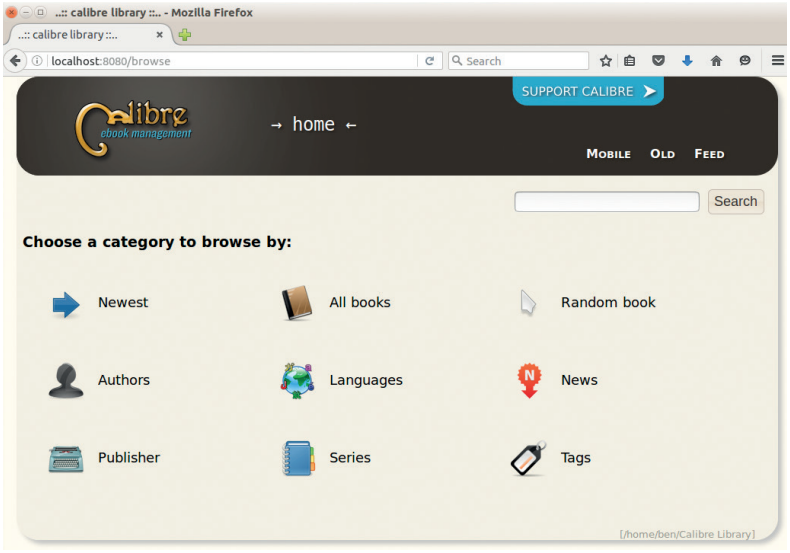
Ben says: Indeed so. I wonder how much other cool stuff is out there mouldering away, lost to the annals or time? Could MI6 let us have the source to some of James Bond's toys, or would that be asking too much?

Thanks, Margaret Hamilton, for your sterling work.

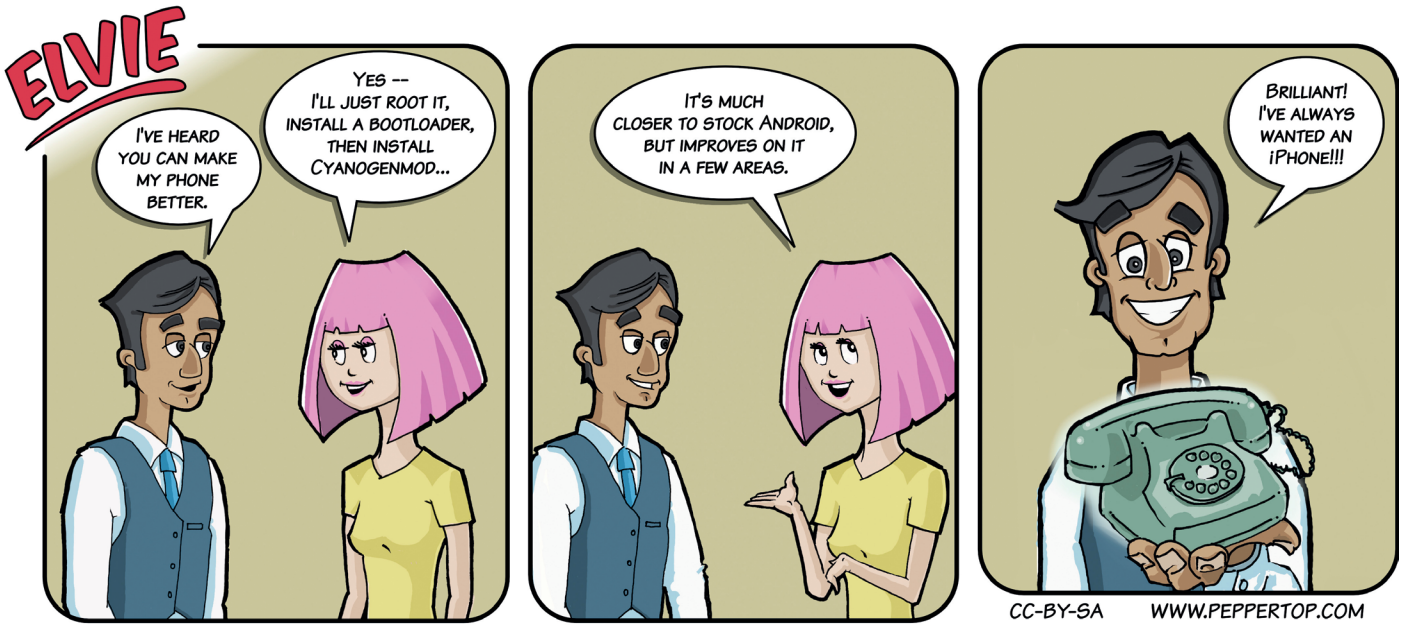
IT'S ONLY WORDS

Cheers for the *Calibre* tutorial last issue. I've been spending far too much time down the rabbit hole of online news what with recent events in UK and US politics, so the ability to filter my own sources has saved me massive amounts of time and get my life back. I've taken back control, as it were. Here's hoping the internet will be safe to go back to when the US election is all over – I like being able to check up on the world without becoming obsessed with it.
Michael Dwyer, London

Graham says: Ah, the internet will never be safe. It's full of rubbish, which is why we like the power to filter off the worst of the nonsense and leave only the best bits, like old Prince bootlegs and NASA source code (**https://github.com/nasa**). Thanks for the feedback!



You decide what's important, with a *Calibre*-curated news feed.



Subscribe

shop.linuxvoice.com

Get your regular dose of Linux Voice, the magazine that:

- Gives 50% of its profits back to Free Software
- Licenses its content CC-BY-SA within 9 months

US/Canada subs prices

1-year print & digital: £95
12-month digital only: £38

Get many pages of tutorials, features, interviews and reviews every month

Access our rapidly growing back-issues archive – all DRM-free and ready to download

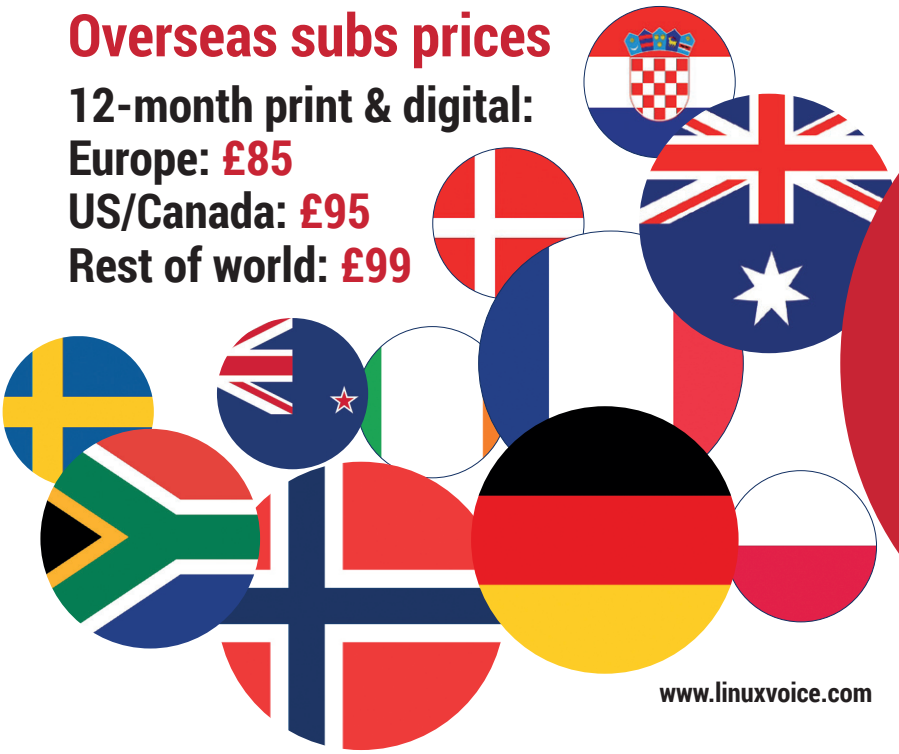
Save money on the shop price and get each issue delivered to your door

Payment is in Pounds Sterling. 12-month subscribers will receive 12 issues of Linux Voice a year. 7-month subscribers will receive 7 issue of Linux Voice. If you are dissatisfied in any way you can write to us to cancel your subscription at subscriptions@linuxvoice.com and we will refund you for all unmailed issues.

All subscribers get access to every single digital back issue – that's about 1,000,000 words of tutorials, reviews and free software hackery at your fingertips

Overseas subs prices

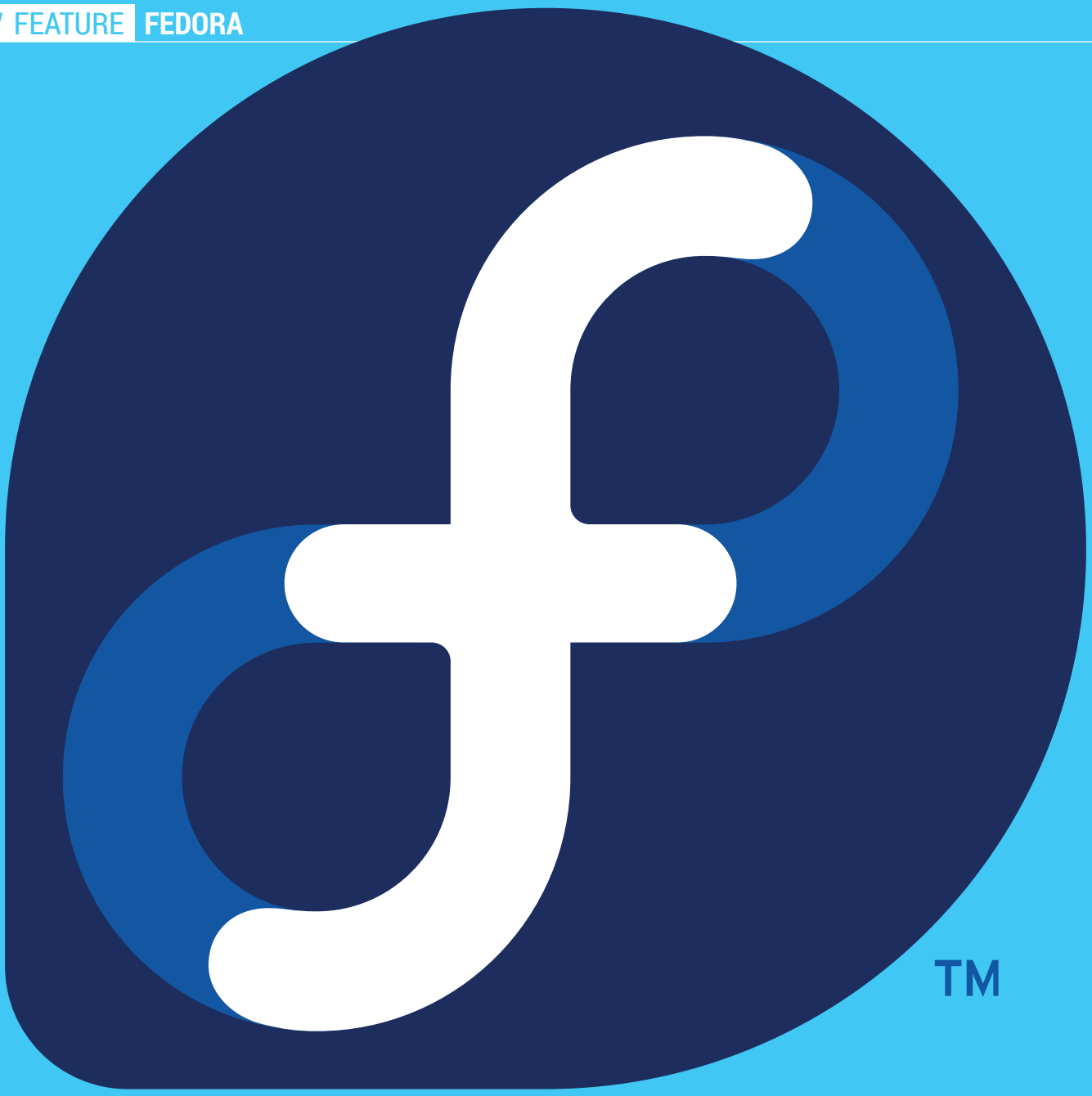
12-month print & digital:
Europe: £85
US/Canada: £95
Rest of world: £99



DIGITAL SUBSCRIPTION*

ONLY £38

*WHEREVER IN THE WORLD YOU ARE – IT'S DIGITAL, SO THERE ARE NO POSTAGE COSTS



FEDORA.NEXT

Speed freak **Ben Everard** dives into Fedora to get the very latest in Linux technology.

Fedora is a fast-paced Linux distribution that's constantly pulling all the latest software from around the Free Software world into a solid distribution. The last sentence is as true now as it's been since the Fedora project released Core 1 back in 2003, but this doesn't mean that Fedora is running the same as it always has. In 2014, the project started to look at how to best position the distribution to take advantage of the latest trends in the Linux world. This fell under the moniker Fedora.next. Two years and four releases later, we can now see just how much effect this change has had. It's transformed the distro from a great option for the Linux desktop to one that also perfect for cutting-edge server deployments. Join us as we delve deep into the heart of Fedora to find out what's going on.

FEDORA.PAST AND FEDORA.NEXT

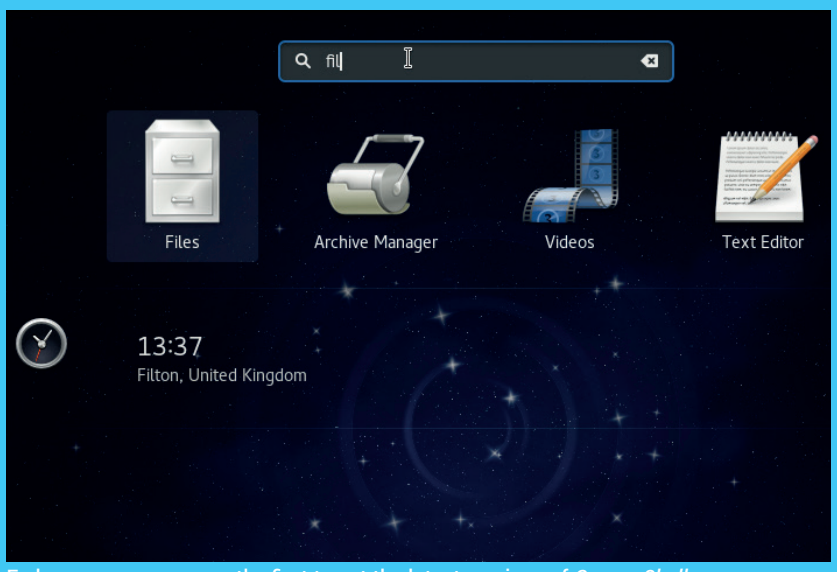
Where we're coming from and where we're going

Things change quickly, and if you're a Fedora user you typically have to upgrade your OS every year or end up on an unsupported version. For most of its life, this means that Fedora has been more suited to desktop use than server. The standard Linux server stack – LAMP – was already mature by the time the first release of Fedora came out, while the Linux desktop environments are still rapidly evolving, so it's only on the desktop that getting the latest software (and putting up with the issues of a changing distro) actually gains you features. For users wanting a more stable environment, the base of the distro, once it's matured in the crucible of Fedora, goes on to form the core of Red Hat Enterprise Linux (RHEL) and CentOS. These distributions take the heart of Fedora and build it into a release that changes far more slowly.

This split between rapidly moving desktop distributions and slower moving server distributions served the Fedora community well for a long time. However, since the early 2010s, there's been a rapid evolution in server technology mostly driven by the desire to serve ever more rich user experiences to ever more people on internet-connected devices. This push has led to new databases, web servers and even paradigms of controlling access to an operating system kernel (containers and lightweight VMs). With these new technologies coming along – and evolving rapidly – the stable server Linux distros such as RHEL and CentOS struggled to keep up with the expectations of modern system administrators.

If you're a Fedora user you typically have to upgrade your operating system every year

The Fedora.next project, launched in 2014, aims to enable Fedora to move into this new technological space while still catering to desktop users. The most obvious outcome of Fedora.next is that there are now three different versions of Fedora: Desktop, Server and Cloud. Each of these is tuned to the needs of different users in the modern technological landscape. We'll look at these over the next few pages to see how they fit together.



Fedora users are among the first to get the latest versions of *Gnome Shell*.

FEDORA FOR THE AGES

25 AUGUST 1991
Linus announces the Linux Kernel on comp.os.linux.

15 SEPTEMBER 1993
First version of Debian (0.01 pre-alpha) released.

12 NOVEMBER 1994
Red Hat version 1 (Mother's Day) released.

12 JULY 1998
KDE version 1 released.

MARCH 1999
Gnome version 1 released.

2002
Fedora Linux starts as a volunteer project to package extra software for Red Hat.

6 MAY 2002
First version of RHEL (2.1) released.

31ST MARCH 2003
last version of Red Hat Linux released (version 9 aka shrike).

5 NOVEMBER 2003
First release of Fedora Core (version 1 aka Yarrow).

18 MAY 2004
Fedora Core 2 (Tettnang) introduced SELinux.

20 OCTOBER 2004
First Ubuntu release (4.10 aka Warty Warthog).

8 NOVEMBER 2004
Fedora Core 3 (Heidelburg) First release to come with *Firefox* and switches from *Lilo* to *Grub* bootloader.

13 JUNE 2005
Fedora Core 4 (Stentz) ships with *OpenOffice 2.0* and Xen.

20 MARCH 2006
Fedora Core 5 (Bordeaux) drops *up2date* and *rhn-applet* in favour of *Yum* for package management.

24 OCTOBER 2006
Fedora Core 6 (Zod) includes AIGLX to GL-accelerate the desktop.

7 DECEMBER 2006
OpenSuse first release (10.2)

22 JANUARY 2007
Linux Foundation formed.

31 MARCH 2007
Fedora 7 (Moonshine) released without 'Core' in the name.

8 NOVEMBER 2007
Fedora 8 (Werewolf) is the first Linux distro to include PulseAudio.

13 MAY 2008
Fedora 9 (Sulphur) brings KDE 4 to users.

25 NOVEMBER 2008
Fedora 10 (Cambridge) brings support for the ext4 filesystem and LXDE desktop.

9 JUNE 2009
Fedora 11 (Leonidas) includes experimental support for the advanced BTRFS filesystem.

17 NOVEMBER 2009
Fedora 12 (Constantine) comes with a preview of Gnome Shell.

25TH MAY 2010
Fedora 13 (Goddard) has a new Internet install option

2ND NOVEMBER 2010
Fedora 14 (Laughlin) is released concurrently on Amazon's EC2

24 MAY 2011
Fedora 15 (Lovelock) includes the new Gnome 3 desktop.

8 NOVEMBER 2011
Fedora 16 (Verne) boots using *Grub 2*.

29 MAY 2012
Fedora 17 (Beefy Miracle) introduces **/run/media**.

15 JANUARY 2013
Fedora 18 (Spherical Cow) supports Secure Boot, and offers Mate and Cinnamon.

2 JULY 2013
Fedora 19 (Schrödinger's Cat) replaces *MySQL* with *MariaDB*.

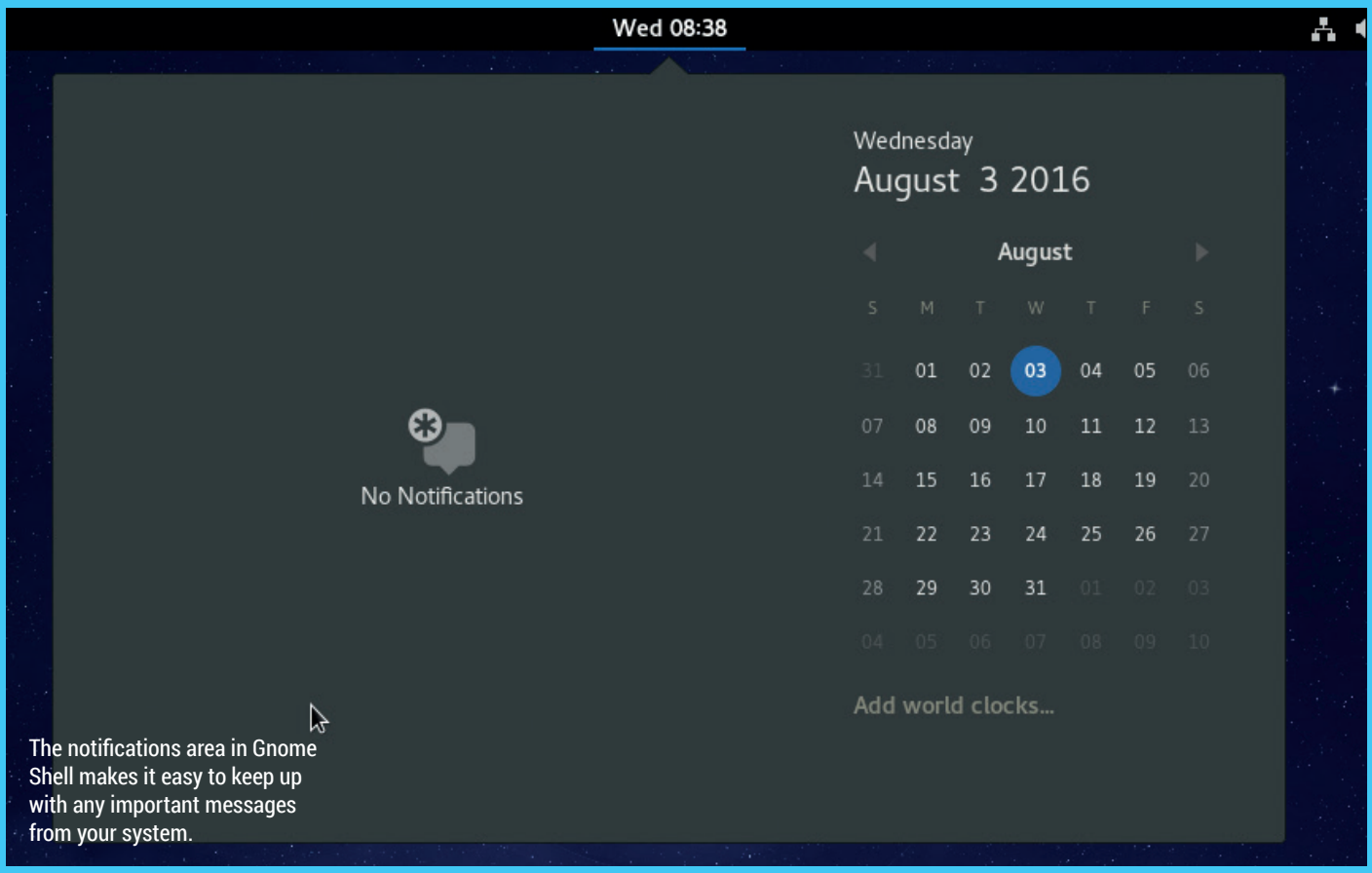
17 DECEMBER 2013
Fedora 20 (Heisenberg) has ARM as a primary architecture.

9 DECEMBER 2014
Fedora 21 is the first release without a codename

3 NOVEMBER 2015
Fedora 23 comes with *LibreOffice 5*.

FEDORA WORKSTATION

Fedora continues to dominate on the desktop



The notifications area in Gnome Shell makes it easy to keep up with any important messages from your system.

The workstation release of Fedora boots into a live Gnome Shell (see boxout for other desktops). Fedora 24 brings this bang up to date with version 3.20 of the desktop environment, so you get all the latest features. You can see our full review of this in the previous issue of this magazine, but our favourite features are the improvement to the help system and a more polished interface.

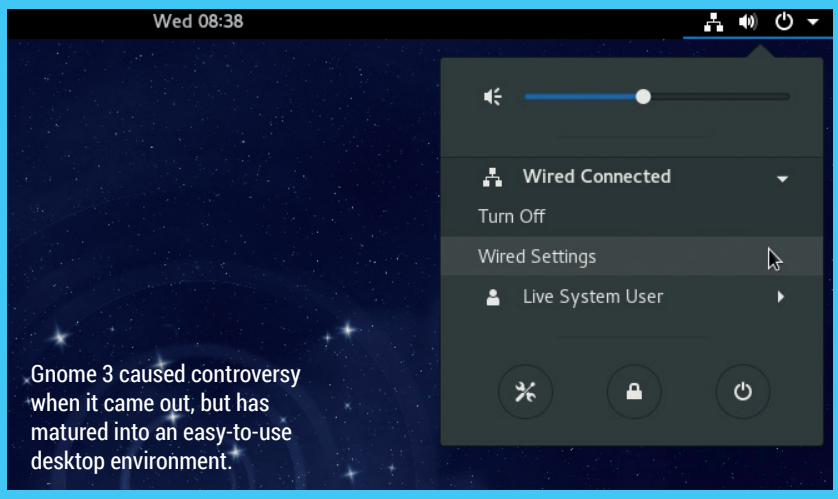
Also new with Fedora 24 was the Flatpak packaging format, which makes it easy to ship desktop applications packaged into containers for security and ease of distribution across multiple distros.

For those of you seeking the ultimate desktop graphics performance, Fedora is probably the distro that best supports Wayland. Although it's not yet the default graphical server, you can enable it. The Wayland experience is getting better with every release, and many users now use it on their normal system without problems.

Spins

Spins are Fedora distributions that come with a different set of default software to the standard Workstation release. Basically, they're for people who want a desktop environment other than Gnome. Here are our favourites:

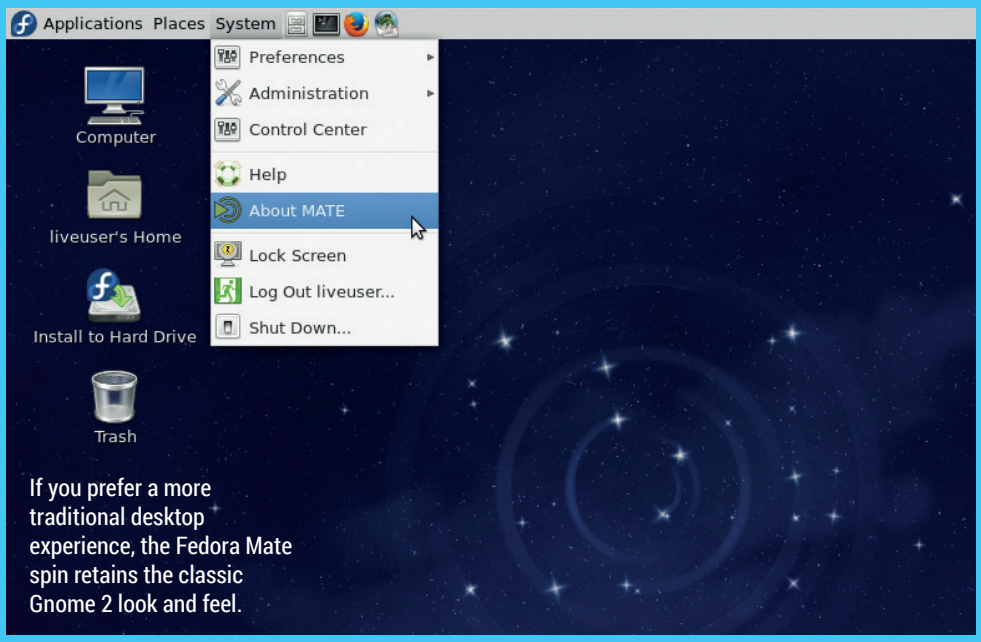
- **KDE** There has been competition between Gnome and KDE for almost as long as there have been Linux desktop environments. Gnome may be the default on Fedora, but KDE is the first option on the spins page.
- **LXDE** Don't wast computing power on a flashy desktop – save it for real work instead. LXDE is a perfectly functional desktop without any of the bells and whistles that waste CPU cycles, making this spin great for older machines.
- **Mate** When it comes to the traditional desktop metaphor, nothing is better than Mate, a desktop forked from the now defunct Gnome 2.
- **SOAS** The least familiar of the desktop spins, Sugar On A Stick is a desktop designed for children initially used by the One Laptop Per Child project.



Gnome 3 caused controversy when it came out, but has matured into an easy-to-use desktop environment.

None of the things we've mentioned so far are exclusive to Fedora, but they are all technologies that are rapidly changing, and Fedora users are well placed to benefit from them before users on other distros. This adoption of technologies at the cutting edge is the real advantage of Fedora rather than any particular piece of software that's unique to the distro.

Fedora is the fastest moving of the non-rolling release distros. Arch users have the very latest software constantly available, but the price they pay for this is not being able to easily predict when big updates come. With Fedora, you'll only ever be a few months behind Arch, but you can know exactly when things will change.



If you prefer a more traditional desktop experience, the Fedora Mate spin retains the classic Gnome 2 look and feel.

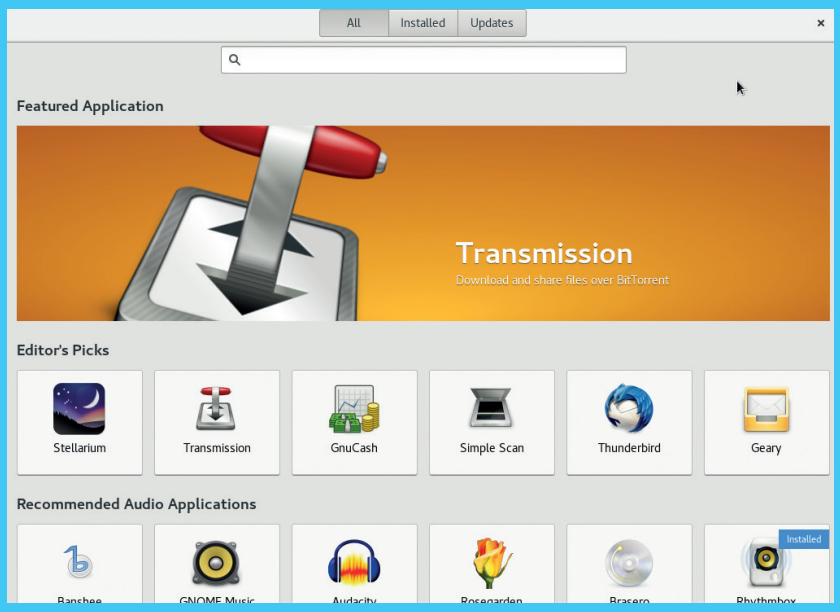
All aboard

When it comes to picking a distro, the community behind the project is important. An active community means that more software will be packaged and bugs are more likely to be fixed, while an open community means that you can get involved and help shape the direction that the distro goes in. The Fedora community is both active and open with most of the activity happening on IRC and mailing lists. If you want to know what's going on with the project, you can browse through the past IRC meetings, as logs are recorded at and posted online at **https://meetbot.fedoraproject.org**.

An active community means that bugs are more likely to be fixed

A significant, and often overlooked, advantage to Fedora for sysadmins and other people who work in operations is that it's the upstream distribution for RHEL and CentOS – the underlying system in Fedora today will be the underlying system in these server distros tomorrow. Having a rounded set of Linux skills means being familiar with multiple distros, but if you're using a RHEL-derived distro professionally, using Fedora Workstation will help you stay ahead of the game.

There is no perfect desktop distro for all people, but Fedora is a great option for anyone who wants a cutting-edge distro with an active community, especially if they want to use enterprise distros as well.



The Gnome Software tool makes it easy to find and install whatever software you need.

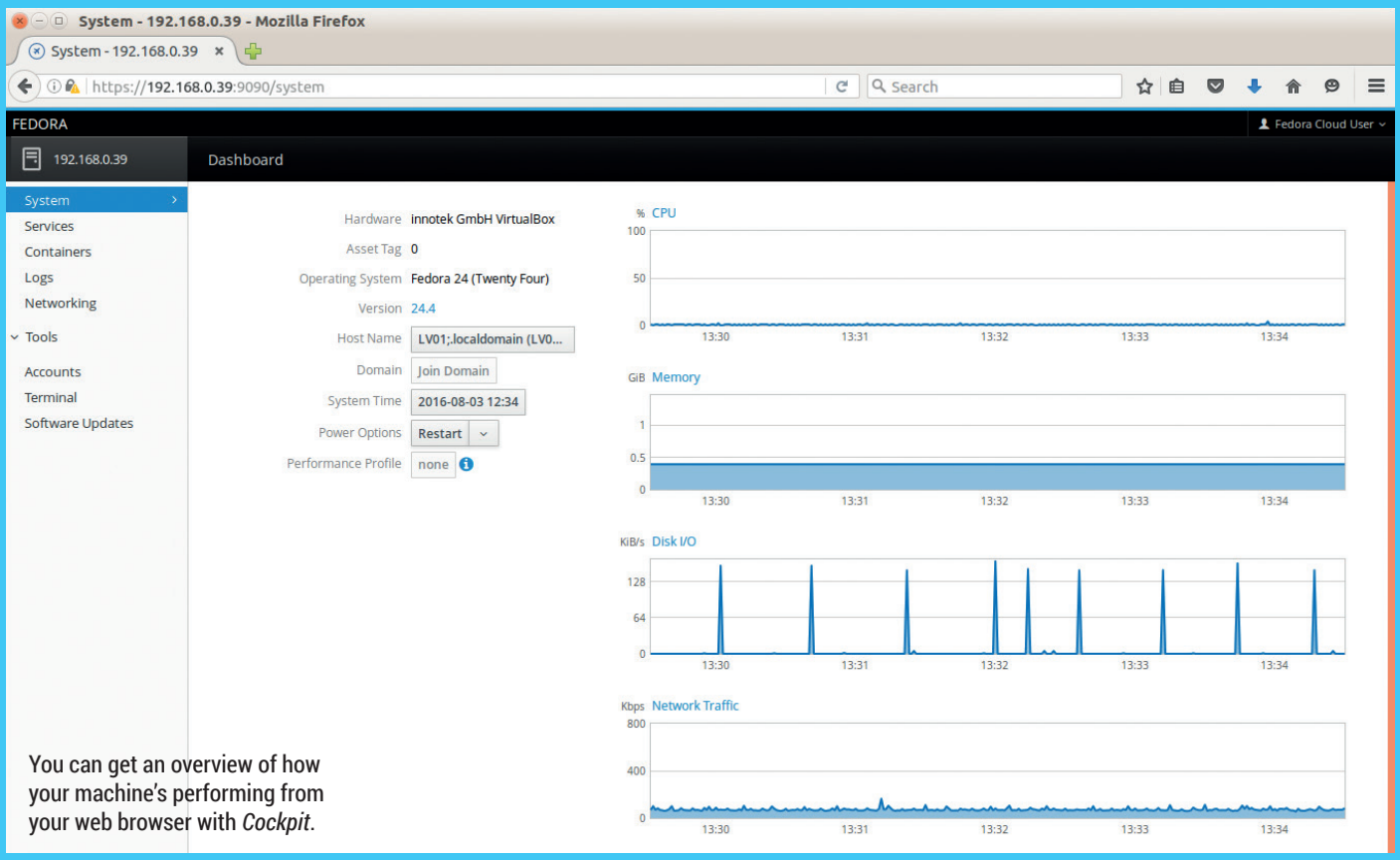
Labs

Labs are collections of software curated for a particular use. To make it easy to get started in a particular area, you can download Fedora releases with all the software already installed, or you can add a lab to an existing Fedora install.

- **Scientific** If you're a student or otherwise working in the scientific community, this lab will pull together all the open source software you need, including *iPython*, *R*, *Latex* and *Gnu Octave*. Spend time probing the mysteries of the universe, not searching through the repositories for the software you need.
- **Security** Security is critical to every tech project, and achieving a good level of security means having a good range of security tools to hand and knowing how to use them. Whether you're just getting started with IT security or are a seasoned pro, the Fedora Security lab is one of the easiest ways to get the software you need.
- **Games** If your machine is more for leisure than work, this is the lab for you. Packed with the best open source games (no Steam here), you're ready to play as soon as you start up. With the live version, you can introduce others to the joys of free software gaming.
- **Design Suite** Free Software has close ties with the design world, and slowly but surely, many designers are reaching for open source as a replacement for the Adobe *Creative Suite* that has been the industry standard for years. The Fedora Design Lab brings all the software you need – whatever your creative medium – into your distro.

FEDORA SERVER

Cutting edge software to help your business run smoothly



You can get an overview of how your machine's performing from your web browser with *Cockpit*.

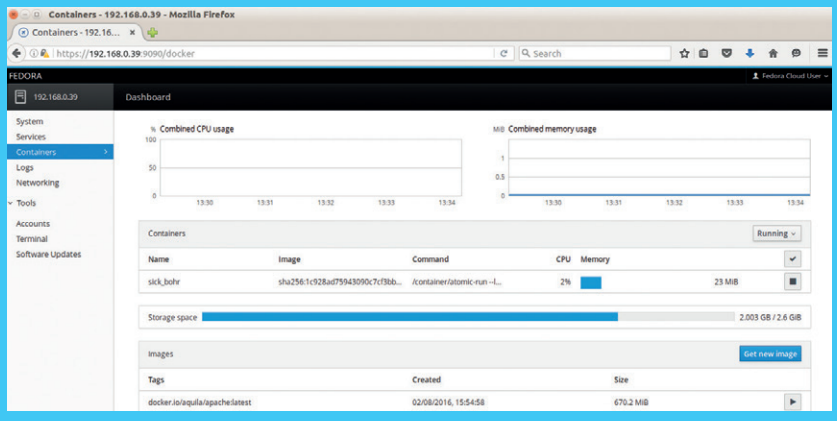
Let's be honest, if you're looking for a server distro, Fedora probably doesn't spring immediately to mind. The though of having just a year or so's support is enough to make most sysadmins' toes curl.

Since the creation of Fedora Server (in version 21 of the distro), the project have thought about what makes a great Linux server, and how to apply that to a fast moving distro. As well as all the usual back-end work on making the distro reliable under heavy loads, and workable through upgrades, there's been work put into making the distro easier to set up and manage.

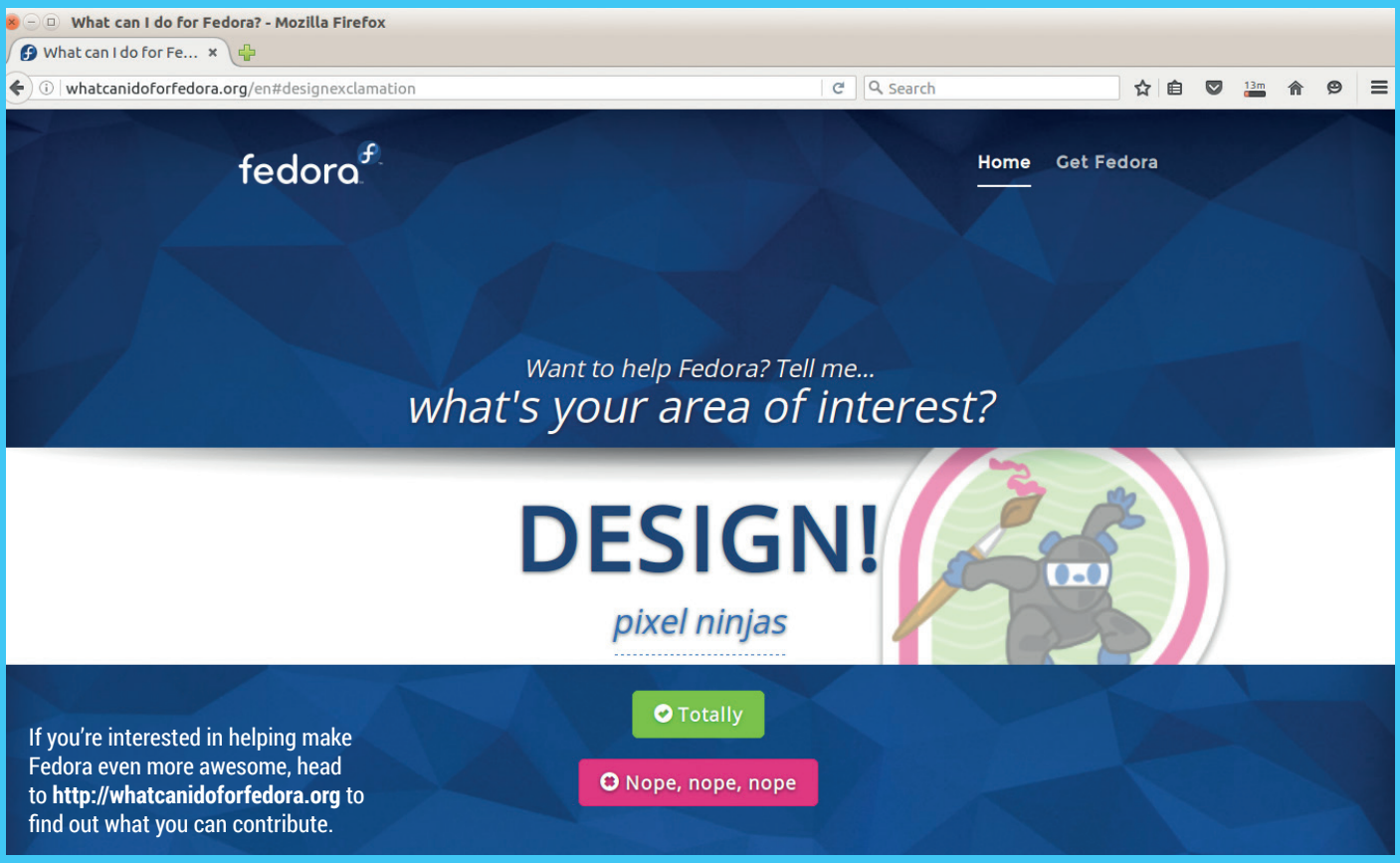
While a standard Linux distribution is a very flexible thing that can be turned to almost any computational tasks, there are a few things that servers are very commonly used to do. Despite these common tasks, Linux system management has traditionally been a little disjointed. You had to repeat the same tasks to get your servers running that thousands of other people did: you had to install the software you needed, configure it to behave as you wanted, then configure any system-level components (such as firewalls) as necessary. This gave you complete control, but could be a little complex, especially for people only managing a couple of servers who aren't full-time system admins, or Windows sysadmins who don't

What can I do for Fedora?

On of the great aspects of Fedora is the community. Although it's sponsored by Red Hat, Fedora is really created by volunteers coming together to work on a great distro. These volunteers come from all walks of life and have a huge range of skills – and they could include you. If you like the idea of helping make the next version of this OS a little more awesome, the Fedora project would love to hear from you. They're so keen to get more people on board that they've made a simple tool to help you find the area within the project where your talents are best put to use. Head to **http://whatcanidoforfedora.org/en** to find out more. You don't have to be a coder or sysadmin to help (though if you are, that's great) – the project is in need of designers, translators and writers. There's something for everyone!



The *Cockpit* interface provides an easy way to manage *Docker* containers on Fedora.



yet know their way around the Linux system.

Fedora has created roles that enable you to very quickly set up a server to a particular task. For example, with just a single line of code, you can transform your fresh Fedora instance into a running database server that's set up with best practices.

If you're coming to a Linux server for the first time, Fedora will help you get started

In the driving seat

Not everyone running a server is comfortable using the command line. This might sound like heresy in a Linux magazine, but it's true. The terminal interface is hugely powerful and with practice can reduce complex tasks to a single line, but it can take a long time to learn to use well. It's a little excessive to expect someone to become familiar with *Bash* just to keep a single server running. Fedora has built *Cockpit* – a web-based management interface – to make life easy for system administrators. You can install it with a single command and then have the ability to tweak your system from your web browser. You can control system services, configure the network, launch containers, update your system and more with just a few clicks of the mouse. If you're a seasoned system administrator, this probably won't make things easier for you as can do all this from the command line just as quickly, but if you're coming to a Linux server for the first time, it will help you get started.

We'll be honest, Fedora Server isn't for everyone. If you've already got a swarm of servers running, and are happy with the process of managing them, it's unlikely that there's anything in the plain Fedora Server distribution to tempt you away from your current setup. However, if you're new to Linux servers, some of the new features could make it a lot more accessible.

So far, we haven't mentioned the software. Fedora is well known for bringing the latest software to the desktop, and the server version also runs on the latest code from upstream projects. There are a few new server projects that are currently iterating really quickly, such as *Docker* and *Kubernetes*. If you're using one of these on a more traditional Linux server distro, you may find that you have to manually keep the software up to date. Fedora Server might seem like a great solution here, but the Fedora project has singled this area out for its own version of the distro – Fedora Cloud.

An alternative route to Fedora on the server

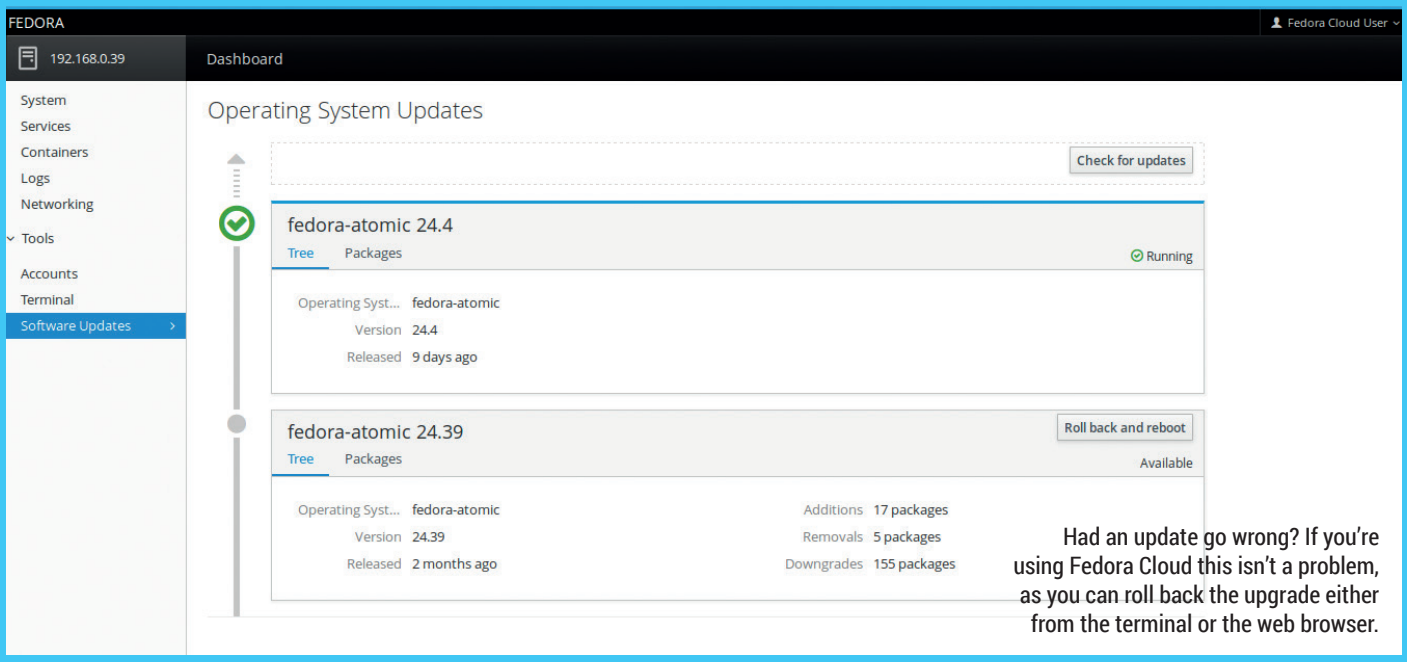
You may be reading this section on Fedora for the server and thinking that having cutting-edge software on a server would be great, but you don't want to have the underlying distro change every six months. It turns out that the Fedora project themselves had this want with the infrastructure they use to run the project.

Their solution was to get the latest version of common server software and package it for RHEL. This way, they got the benefits of the stable base of RHEL, but also the benefits of faster-moving packages. Seeing that other people wanted this solution, the Fedora project grouped these new packages into a repository called Extra Packages for Enterprise Linux (EPEL). Although these repositories are maintained by Fedora, they're for installation on RHEL, CentOS, Scientific Linux and Oracle Linux.

This is something that we at Linux Voice appreciate, because our servers run on CentOS with EPEL additions. Like many others, we find the balance between stability in the core OS and having up-to-date software in the repositories the best solution for our needs.

FEDORA CLOUD

Master your containers for easy deployment and scalability.



Had an update go wrong? If you're using Fedora Cloud this isn't a problem, as you can roll back the upgrade either from the terminal or the web browser.

Fedora Cloud images

Fedora Cloud comes as image files that can be put directly on to virtual machines rather than ISO files that need to be installed. The obvious problem with missing the install step is that it doesn't catch the useful information that users typically entersuch as such as login credentials and any necessary network details. You can pass all this information to the Fedora Cloud instance using an **init.iso** image. This is a CD image that contains two text files detailing the data that Fedora needs. The most basic setup is the following. A text file called **user-data** containing:

```
#cloud-config
password: fedora
chpasswd: { expire: False }
ssh_pwauth: True
```

and a file called **meta-data** containing:

```
instance-id: iid-LV02;
local-hostname: LV01;
```

With these files in place, you can create the ISO image with the following (run in the same directory as the above files):

```
genisoimage -output init.iso -volid cidata -joliet
-rock user-data meta-data
```

You'll now need to download either the Base or the Atomic Host image. Whichever you pick will come as a **QCOW** file that can be booted up in *Qemu* (or *KVM*), but we ran ours in *VirtualBox* as it makes it more straightforward to sort out networking. Before you can use *VirtualBox* though, you need to convert the image into a VDI file with:

```
qemu-img convert -f qcow2 <downloaded-
filename>.qcow2 -O vdi <output-filename>.vdi
```

You can now create a virtual machine with the VDI image as the hard drive and the ISO file in the CD drive. We used Bridge networking (in Settings > Network > Bridged Adapter) to simulate the virtual machine as being on a new network interface. Starting the machine will take you to a login prompt where you can log in with fedora/fedora.

Before we delve too far into the Fedora Cloud images, let's stop for a moment to work out what we mean by Cloud. It's been a computing buzzword for over 10 years, but it's still used by different people to mean different things. At its heart, cloud computing is any computing that can easily be done on different remote machines. This can be true on a variety of levels. At the top level, it's about applications that can run anywhere. Take, for example, a cloud office suite such as Google Docs – this runs in your web browser but the back end could be anywhere; it doesn't matter to you as long as you can access it through **https://docs.google.com**.

A level lower, you can access a software platform that could be an operating system or a programming language environment. The lowest level is providing virtual computing power (such as virtual machines) though the cloud.

The essential part of all this is that the user pays for a specific service they want rather than a specific piece of hardware or software. The abbreviations of cloud software often end in AAS (As A Service) to denote this. For example, an online office suite is Software as a Service (SAAS); a software platform available over the cloud is a Platform as a Service (PAAS); whereas a provider of online virtual machines is Infrastructure as a Service (IAAS).

What is the cloud?

Cloud computing is when you contract out a specific part of your computing needs to a third party. This can have huge organisational and cost saving advantages because it means that one company can specialise in one thing (such as running online office suites) and do so highly efficiently, while another can focus on its core business rather than try to keep their office suites running and updated.

The biggest problem with this view of the cloud is that it requires you to hand over much of your data to the third party. If they go bust or get hacked, you're in trouble. In order to solve this problem, large organisations run a private cloud where the IT department provides some facilities to the rest of the organisation in a cloud-like manner. Typically this is a Platform as a service (PAAS) or Infrastructure as a service (IAAS).

Phew! That's quite a lot of detail to get through before we even look at what the cloud release of Fedora does.

The Project Atomic website is the best place to get information about how to use Fedora Atomic Host.

Firstly, let's just discard Software as a Service – it has its place, but it's not the sort of cloud we're talking about. This distro is really about working with a PAAS or IAAS system.

You could use a Fedora Cloud image to control containers running on *Docker* that are hosted on someone else's PAAS; you could build your own private PAAS using Fedora Cloud as the host OS; or you could use Fedora Cloud on an IAAS. There are three versions of Fedora Cloud that suit different uses cases:

- **Base** A minimal Fedora release for creating virtual machines on IAAS.
- **Atomic Host** A Fedora build specifically for running containerised applications with *Docker* and *Kubernetes*.
- **Docker** This release is purely for running inside *Docker*, and you shouldn't usually download it directly. Instead you can get it through *Docker* either directly or by creating a *Docker* file that builds from it.

Whichever you choose, you'll find that they don't come as ISO DVD images as is normal for Linux distros, but as virtual machine hard drive images. This reflects the way they're designed to be used. Rather than installed onto a machine through the traditional install process, they're imaged in a state ready to run.

It's the Atomic Host version of Fedora Cloud that is really bringing next-generation system administration to the Linux world. It's part of Project Atomic (which also spans CentOS and RHEL) and aims to create a platform that makes it as easy as possible to build and deploy containerised applications across a pool of computing resources.

The Atomic Host version of Fedora Cloud is bringing next-gen Sysadmin to Linux

The underlying technology for this is Linux Containers, *Docker* and *Kubernetes*, but Project Atomic hides as much of this as possible behind a single interface (the **atomic** command and the nulecule container specification) that makes it easier to manage.

With all your applications in containers, the base system should always be small and easy to manage. The base system is upgraded using the **rpm-ostree** command, which keeps track of all previous system states so that if you encounter a problem with the upgrade, you can roll back to any previous state. With this roll-back option, keeping up with a fast-moving distro poses fewer risks.

Fedora Cloud offers a radically different server management paradigm that aims to make it easy to manage servers and scale applications. Much of the technology behind this approach is new and evolving quickly, and Fedora is ideal for making make the most of this new approach.

Kubernetes, Docker & LXC

The new technologies that Fedora Cloud is built on can be a little baffling even to people who have been using Linux for a while. Let's take a quick look at these three:

- **Linux Containers** (aka LXC) is the name given to a group of technologies that isolate a set of system resources running on top of a Linux kernel. This effectively gives you the ability to run two or more Linux systems on a single CPU. Only a single kernel runs, but the two systems are so well isolated that the two systems appear to be completely separate from each other.
- **Docker** Using a system of images, users can design containers using Dockerfiles that define what distribution the container should be based on and how it should be configured. These images and Dockerfiles can be shared through a *Docker* registry that makes it really easy to create, modify and share containers for almost any need.
- **Kubernetes** controls groups of *Docker* containers to make it easy to build systems running across multiple machines. Take, for example, a situation where you had 10 physical machines and wanted to deploy a web app that required a web server and a database. Using *Kubernetes*, you could define *Docker* containers for the web and database server, then spin up instances of these containers across your 10 machines and load-balance the system to keep everything running smoothly.

RetroPie

BACK TO THE FUTURE

Turn a spare Raspberry Pi into an awesome retro gaming station. **Mike Saunders** gets misty eyed reliving the classics.

We often hear from readers that many of you now own multiple Raspberry Pis. As they're so cheap, it's not the end of the world if one spends the next few years sitting in a cupboard, but we always have this nagging feeling that any working technology should be doing something useful.

Now, there are obviously a million and one things you can do with a Pi, from home servers through to robotics, but one thing we like about it – especially thanks to its HDMI output port – is that it's very much "plug and play". With the right software on the SD card and a HDMI cable, you can plug it into almost any TV made in the last five years and it just works.

Now you're playing with power

And this is especially useful for emulating classic video games consoles and computers. You can do this on a desktop PC or laptop of course; Linux has a wealth of first-rate emulators for pretty much every platform under the sun. But setting up emulators on a typical distro can be fiddly, and having a bulky PC under your TV (or a laptop that can't be used for another job at the same time) isn't idea.

So the Raspberry Pi is perfect for this: low-power, silent, plug-and-play as mentioned, and with a couple of USB ports for joypads. Over the next few pages we'll show you how to set up a spare Pi using the very, very cool RetroPie distribution, so that you can play old games from the 1980s and 1990s with minimum fuss. Sure, there are some excellent modern games, but very little compares to multiplayer *Super Bomberman*...

SETTING IT UP

Install RetroPie and prepare to beat your old high-scores.

The first thing you need, of course, is a Raspberry Pi. Excellently, all models of the Pi can be used with RetroPie, so you're not just limited to the latest versions – but there is an impact on performance. In our experience, the earlier models (and Pi Zero) do a decent job of emulating the 8- and 16-bit classic consoles such as the Atari 2600 (VCS), NES, SNES, Master System, Mega Drive (Genesis), Game Boy and Game Gear. If you want to try emulating more advanced machines, like the Sega Saturn or Sony PlayStation, a Raspberry Pi 3 is highly recommended – and even then, you'll get mixed results. We recommend RetroPie mainly for 8-bit and 16-bit emulation.

Next, you'll need the usual bits and pieces to get a Pi running and connect it to a TV: a power cable, a HDMI cable, and an SD card (8GB minimum recommended). Then you'll need a USB joypad (or two) to actually play the games – you can pick up SNES knock-offs from Amazon for a few pounds/dollars/euros. Finally, if you want to do some tweaking of your RetroPie installation, it's a good idea to have a USB keyboard around that you can plug in. You don't need it to play games, but you'll need to use it to gain access to certain advanced features.

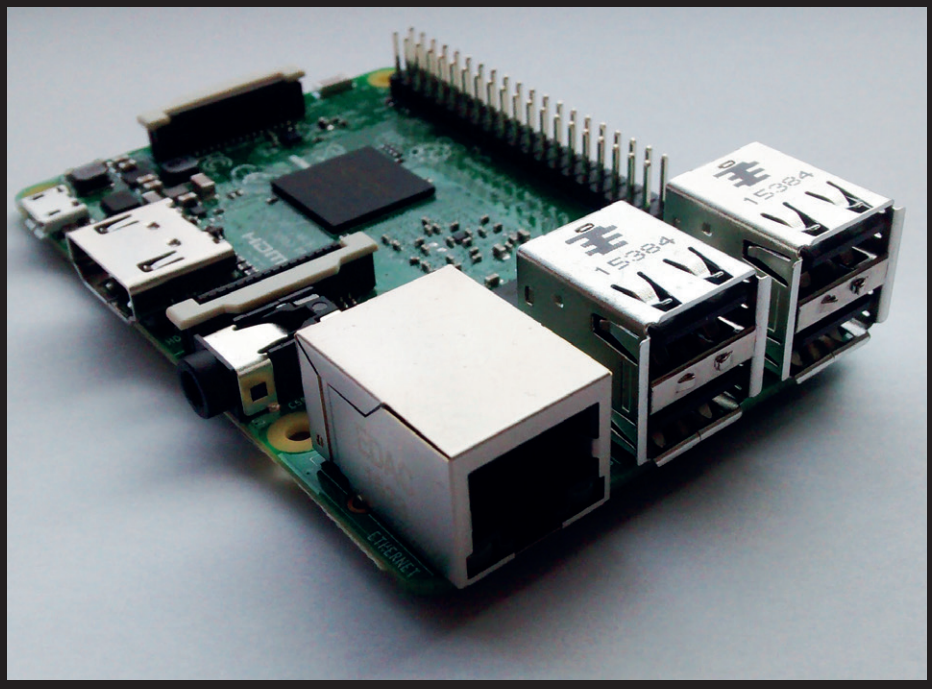
So once you have all your equipment, head over to the RetroPie download page at **https://retropie.org.uk/download**. You'll see that there are two download options: one for older Pis (models Zero and 1), and another for the Pi 2 and 3. Choose which is appropriate for your device, and after a few minutes of downloading you'll end up with a **.img.gz** file containing the filename. This is a compressed Raspbian image, much like the normal ones you install on a Pi, but highly customised for retro gaming.

Play your cards right

Next you need to write it to your SD card. Move the file to your home directory (eg **/home/mike**), then open up a Terminal (command line) window and use **gunzip** to extract the compressed image like so:

```
gunzip retropie-v3.8.1-rpi1_zero.img.gz
```

Note that the filename may be different in your case – and remember you can hit Tab after typing the first few characters of a filename to auto-complete it!



Every Pi model works well with RetroPie, but for the best performance (eg for PlayStation emulation) we'd recommend the beefed-up version 3.

Next, enter **df -h**, which shows a list of disks (or SD cards) currently in use on your Linux distribution, along with how much space they have. Now plug in the SD card you want to use for RetroPie, and run **df -h** again. This second time of running the command, you'll see the difference from the first output, showing which card you've added. Note the SD card name in the list – eg **/dev/sdd1**. We want to access the SD card directly, and not via a file manager, so unmount it like so:

```
sudo umount /dev/sdd1
```

(Change the device name accordingly.) If the output of **df -h** before showed multiple partitions on your SD card, you'll need to unmount those as well.

Now we need to directly write the Raspbian image to the SD card. To do so, we use the name we had before (eg **/dev/sdd1**), but this time removing the number, as we don't want to access any specific partition but rather the whole device. So:

```
sudo dd bs=4M if= retropie-v3.8.1-rpi1_zero.img of=/dev/sdd
```

Be very careful with this command – if you put **/dev/sda** or something else, you could end up overwriting your main drive! If you have any doubts or need more help, see the official Raspberry Pi SD card installation documentation at **www.raspberrypi.org/documentation/installation/installing-images/linux.md**.

Anyway, the above command could take anything from a few seconds to several minutes depending of the speed of your SD card, so go off and make a cuppa. When it has finished, enter **sync** at the command line, then remove the SD card and plug it into your Pi. You're ready to go!

Play it loud!

Now connect your Pi to the TV, plug in your USB joypad(s), and RetroPie will boot for the first time. It will expand the Raspbian installation to completely fill the SD card,

A Raspberry Pi running RetroPie, playing *Sonic 1*, but with a SNES pad connected via a Retrode. What kind of black magic is this?

then reboot and you'll see a shiny splash screen. Once the emulator front-end has loaded, you'll be prompted to press and hold a button on your joypads to configure them.

RetroPie will ask you to assign each button on your USB joypad to a button in the emulators (eg A, B, X, Y – do a web image search for a SNES joypad to get an idea of the standard layout). If your joypad has fewer buttons or you don't want to assign a physical one to an emulated one, just press and hold any button when prompted.

Once the joypad(s) are configured, RetroPie will display a list of emulated consoles and computers – use the left and right buttons on your joypad to navigate between them. As you have no ROMs installed at this stage, you won't have much to play, but you can test your installation by going to the Ports option and choosing *Doom*. If all goes well, you'll be able to play a bit of classic *Doom* and ensure that your joypad is set up correctly.

To exit out of *Doom* – and indeed any emulator – press the buttons you have assigned to Start and Select simultaneously. This will return you to the RetroPie front-end, where you can navigate to other emulators and games. (Press A to go into an emulator, and B to switch back out.) If your joypad wasn't configured correctly, press Start in the main RetroPie screen to open a menu, go to Configure Input, and then go through the process again.

Adding ROMs

Now you'll want to add some ROMs (see the "Thorny issue of ROMs" box for more information on these). If you're savvy with the command line, you can plug a USB keyboard into your Pi, hit F4 to switch to the command line, and then **cd** into the **/home/pi/RetroPie/roms** directory. There you'll see subdirectories for various consoles – most of the names are obvious, but note that **gb** is for Game Boy, **gbc** for Game Boy Colour and **gba** for Game Boy Advance.

You can now copy ROMs into these directories, eg from a USB key. If you're not so *au fait* with the command line, press Ctrl+D to return to RetroPie, and then shut it down by hitting the Start button and then Quit in the menu. Once your Pi has fully shut down, remove the SD card and insert it into your computer.

In your file manager, navigate to the **retropie** partition of the SD card, and go into **/home/pi/RetroPie/roms**. Here you can add ROMs into the appropriate directories as mentioned previously. When you're finished, unmount the SD card, pop it back into your Pi, reboot and enjoy playing the classics!

RetroPie has many advanced features including the ability to scan for ROMS, so visit **https://retropie.org.uk** for documentation and forums where you can get help and share ideas. And if you discover an absolute gem from the 80s or 90s that everyone should play, drop us a line!

THE THORNY ISSUE OF ROMS

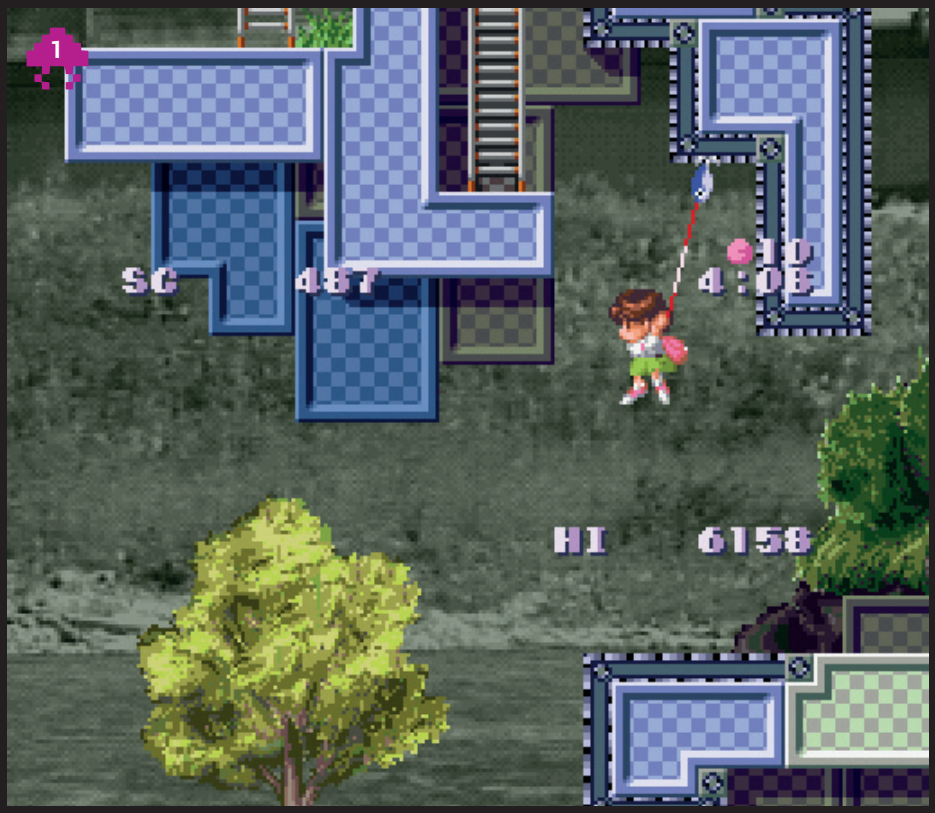
An emulator isn't much fun without any games to play on it. But where do you get them? If you're new to emulation, it's important to understand what the term "ROM" (read-only memory) means. In this context, it's a file that represents the contents of a video game cartridge (or floppy disk or CD-ROM). It essentially contains the very same bits and bytes of the original game.

Now, the internet is awash with ROMs, but their legality is dubious. Some people argue that downloading a ROM for a game you already own is fine – after all, you have a right to own a backup copy in case your game breaks. Others say it's still a copyright infringement. Many people would argue "Ah, these games are so old, it's not like the original vendors are losing any money." But some companies are still selling old games – look at Nintendo with its new NES Classic Edition.

So play it carefully. If in doubt, buy a machine to directly extract ROMs from cartridges you own (eg a Retrode). Or play the many home-brew ROMs created more recently by fans of classic consoles – most of those games are released for free and their developers actively encourage copying and distribution.

FIVE OF THE BEST

Mike's selection of relatively obscure but unmissable retro games.



1 Umihara Kawase (SNES) From the outside, this looks like a traditional jump-and-run platformer, but it has a twist: your character welds a fishing rod, and you can fire its hook onto other platforms to swing around, Spiderman-style. It makes you think about level structure in a wholly different way, and there's some corking music to accompany your antics too.

2 Unirally (SNES) Known as *Uniracers* in the States, and created by DMA Design (of *Lemmings* and later *Grand Theft Auto*) fame, *Unirally* is a side-scrolling racing romp in which you perform stunts on a unicycle at breakneck speeds. It's shallow, but tremendous fun to play – especially in split-screen two-player mode.

3 Elite (NES) We've talked about *Elite* (and *Frontier*) plenty of times in Linux Voice, but you may not be aware that it was ported to Nintendo's mighty NES. This was a huge challenge for a machine that had character-cell graphics – 3D images have to be mapped onto sprites. But it works well, with a clever joypad-based input system and top-notch battles.

4 Kaeru no Tame ni Kane wa Naru (Game Boy) This was only ever released in Japan, but the ROM has been translated by hardcore fans and is playable in English. It's based on the engine used in *Zelda: Links Awakening*, and it's also an great role-playing game (although very tough in places).

5 Dragon's Revenge (Mega Drive/Genesis) Most pinball games tend to be shallow affairs with pretty sounds and shiny graphical effects, but not much else to keep you glued to the joypad. Dragon's Revenge is way better: a fantasy world, boss levels, bonus balls and other tricks make it one of the superior pinballers. LV

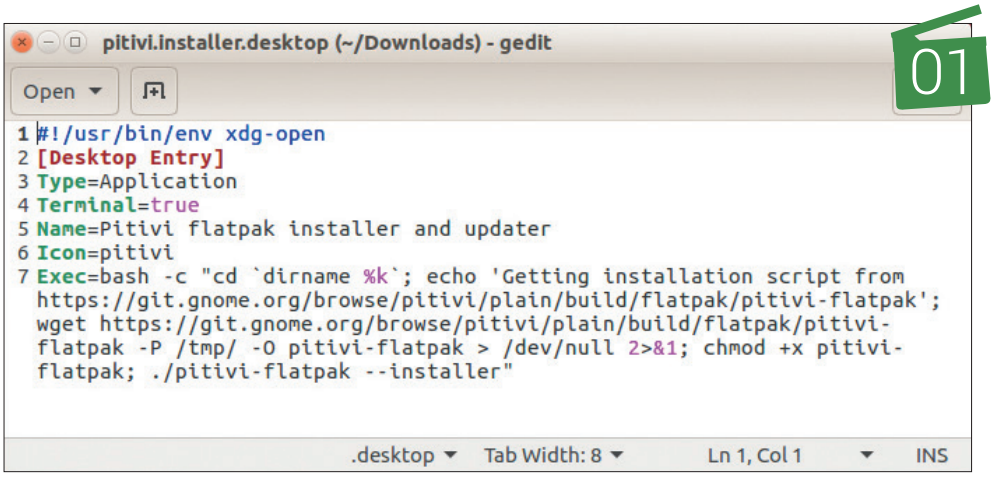
SECRETS OF PITIVI

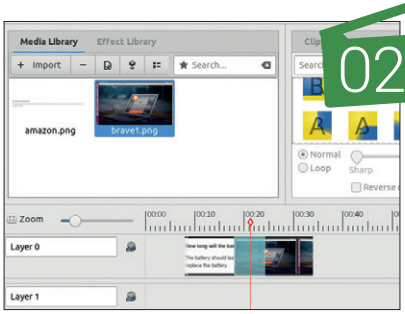
Get the most out of your motion pictures with open source video editing

Fast internet speeds and the proliferation of smartphones mean that we're creating and sharing more video than ever before. As a result of this, video editing software is becoming more popular for stitching together GoPro camera footage of sports, mobile phone footage of current events and, of course, videos of cute cats.

This month, we're taking a close-up look at one of the most popular open source video editing options, *Pitivi*. Although there is some serious competition, this is our favourite of the open source video editing options. *Pitivi* is a powerful and rapidly developing piece of software that has really started to shine in the last couple of releases. The stability problems that plagued it for years have now mostly been consigned to the past, and the new-look *GTK 3* interface makes it a pleasure to use.

We've been using *Pitivi* for a few years – here are our eight favourite features.





01 Flatpak

Installing software in Linux is easy, provided your distro's repository has the version you want. If you're after the very latest version though, things can get complicated. The *Pitivi* developers have packaged the latest version up with Flatpak to make it easy to install on any distro that supports this format. We were able to download and install the software on Ubuntu with just two commands, which makes it just as easy as **apt-get**.

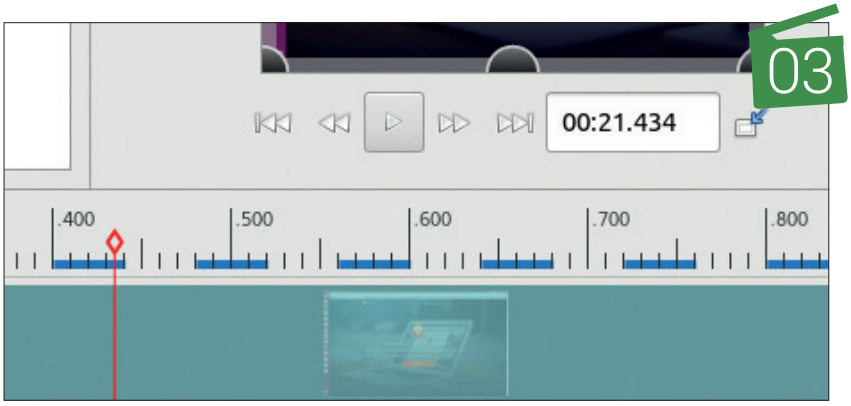
02 Transitions and filters

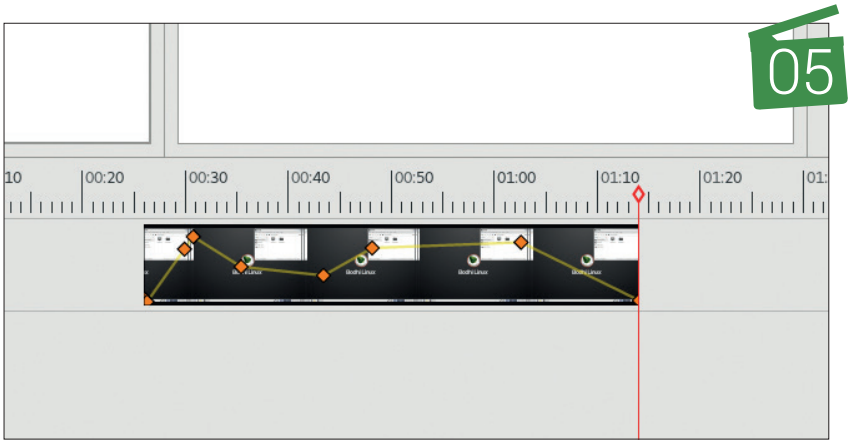
Good use of graphical effects requires an eye for style and plenty of creativity. *Pitivi* can't help you with these, but it does provide vast numbers of options that are easy to add to your movie. There are over 70 transitions and over a hundred effects for you to play with – this is more than enough to add a professional look to your projects – just don't add all of them at once.

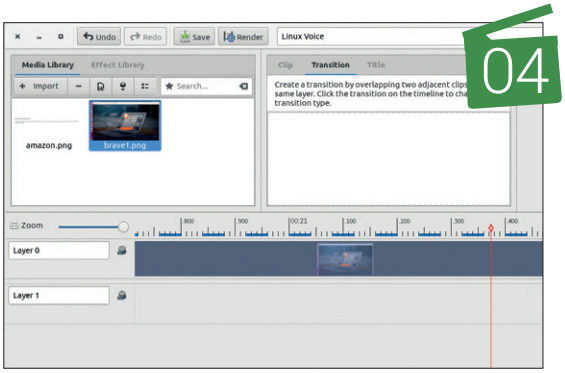
> Pitivi has over 70 transitions and over a hundred effects for you to play with – more than enough to add a professional look to your projects

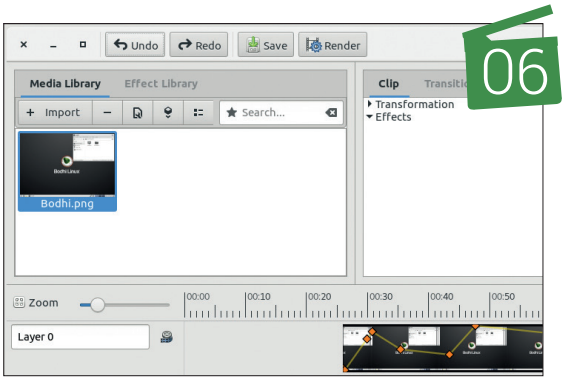
03 Nanoseconds

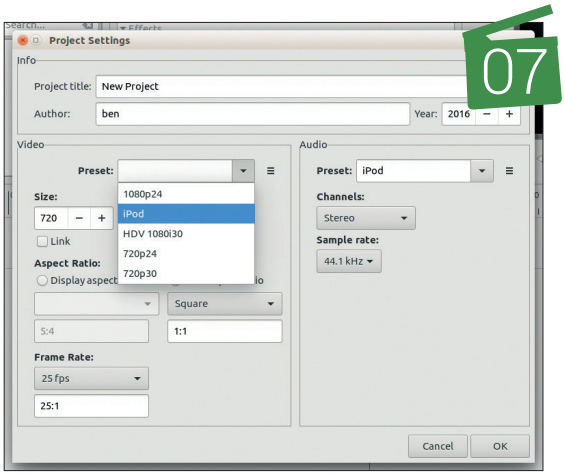
There are a lot of components to a video – multiple tracks, effects and audio are all pulled together to make the final video and these all have to happen at the same time to get the right result. If this synchronisation is off by just a small amount, the result is a video that looks amateur. Just how accurate does this timing have to be? Well, precision to

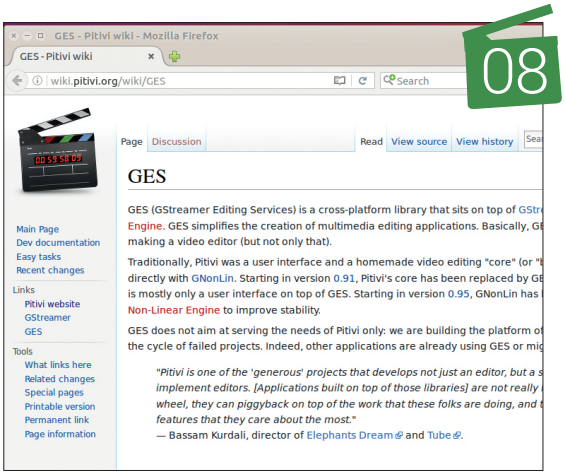












the nanosecond is more than most people need, but it's better to have too much precision than not enough.

04 Background processing

Video processing takes a lot of processor power. This can mean delays as you wait for things to run through. *Pitivi* does as much as possible in the background, which means that it enables you to continue to work on your project while the CPU is churning through your data. This means that the heavy processing demands don't slow you down.

05 Keyframes

When linking effects to a section of video, you need a way to place events at a particular point in time. Using keyframes you can tag moments in the video and use these to inform the effect or transition. These keyframes enable you to precisely control the way things like fades, movements, pans and cuts happen, and this stays the same if you move the video to another point in the project.

06 Full undo/redo

We all make mistakes, and we also make intentional edits that end up not looking as good as we thought they would. *Pitivi* has a full undo and redo system so that you can scroll backwards and forwards through the changes you made to get the project back to the exact state you want it in before you start to make more edits.

07 Render to profiles

There are hundreds of ways to watch a video – just a few of the most common are: on a television in your living room, (in both normal and HDTV flavours), on a phone on the train on the way to work, on YouTube and in a desktop media player. All of these have different attributes and show differently rendered videos best. *Pitivi* can render to different profiles so you can output your project so that it'll be at its best on different devices.

08 Scripting

The *Pitivi* project has developed both the front-end user interface and a video editing core called *Gstreamer Editing Services* (*GES*). This back-end is independent of the user interface and enables technical users to build scripts and software that can access the huge editing power of *Pitivi*. By building their software in this way, the *Pitivi* project has vastly reduced the difficulty of incorporating non-linear video editing into other projects.

THE BIRTH OF LUMO

Discover games development with the secret diary of **Gareth Noyce**, creator of the superlative isometric platformer, Lumo.

We loved the game *Lumo* when we reviewed it a couple of issues ago – it's a tough isometric puzzler with 21st century graphics and the soul of a game that should take 15 minutes to load off a cassette. What's even more remarkable is that, like many of the 1980s games that helped to inspire *Lumo*, it's mostly the product of a single developer: Gareth Noyce. And following in the footsteps of 80s bedroom programmers like Andrew Braybrook and Jeff Minter in magazines like *Zzap!64*, we thought we'd ask Gareth if he'd consider writing a diary of his development process. And remarkably, he agreed...

Gareth who?

Gareth Noyce started his career with Climax Studios Solent, first as QA and then as a designer on the Xbox RPG, *Sudeki*. He moved to Climax Studios London as a producer before he partnered with Gary Liddon (yes, *Zzap!64*) as a co-founder of Xen Services, and spent several years as a contracted producer, working on *Crackdown*, *Too Human*, *PGR3*, *Fable 2*, *Space Giraffe* and the Xbox 360 visualiser, *Neon*. In April 2008, Gareth, Gaz Liddon and Billy Thomson (ex DMA Design & Realtime Worlds) formed Ruffian Games in Dundee, where Gareth worked as development director.

Over the course of the next five years Ruffian would ship another five titles for MS: *Crackdown 2*, *Nike+ Kinect Fitness*, *Kinect Sesame Street*, *Season 2*, *Kinect Star Wars* & *Kinect Playfit*, and was heavily involved in Crytek's *Ryse* during pre-production, as well as several unannounced and canned projects.

In April 2013 Gareth moved to Finland and began teaching game design, Unity3D programming and business & production courses at TAMK Univeristy. *Lumo* was born toward the end of 2013 and is the first project to come out via Triple Eh? Ltd. In addition he provided a music track for the Llamasoft Playstation Vita hit *TxK*, as well as music for an unannounced PS4 VR game.

Lumo's play mechanic and graphical style was set very early, as these shots from the first couple of days of development show.

29 April 2013

I arrived in Finland to start up a new dev-studio, the idea being to make mobile games and learn from the vibrant development community that had sprung up around companies such as Rovio and Supercell. That didn't quite go to plan. In fact, I ended up doing something completely different and going it alone. *Lumo*, my first independent game, was the result; a modern take on the classic 'Isometric Arcade Adventure' genre (*Head Over Heels*, *Amaurote*, *Knightlore* etc) that was a mainstay of the 1980s gaming scene. Development took around 2.5 years, part-time, and the final result was released in May 2016. This is, roughly, how it happened.

November, 2013

For the last three months I've been working on an RTS prototype – think *Advance Wars*, and you'd be close – aimed at the tablet market. Something about it seems like a great fit between game and form-factor, but it's becoming increasingly clear that I'm not going to be able to finish it and release it as a free-to-play game. I've not got the money for user acquisition (marketing by another name) and I'm not sure I can support it with enough content after release. It's depressing and my constant mooching about is getting on my girlfriend's nerves: "Why don't you just make something you want to do, and forget about the mobile stuff?!" I think she has a point, but I'm still not sure what I can do on my own that I can finish to a high enough quality.

I'm at a party, a little worse for wear, idly chatting to Ste Pickford over Twitter. He's one of the Pickford brothers, responsible for many classic games, and we're watching speed-runs of people playing *Equinox*, when it hits me... I've always wanted to make a game like *Head Over Heels*. It was the first game that I actually owned, so there's a nice circularity to it. They're reasonably small and I could probably do the whole thing on my own. Eureka!

When the hangover subsides I get up and start work on a quick prototype. It instantly clicks. I think this is it. I show my girlfriend and make a quick video to share (**https://www.youtube.com/watch?v=uRh_yDp3XF4**) with Ste, who both make encouraging noises. *Arse Over Tit* is born, and I have my project!

December, 2013

I'm nearly full-time on *Arse Over Tit*. The university where I teach is nearing its winter break and the short consultancy job I'd picked up has finished. The design ideas have settled very quickly. I know that I want to ape the kind of world that *Head Over Heels* presented, so I've settled on the location being a big, scary castle, with two or three distinct zones, each with their own art style and set of toys for the player to discover. I want the player to feel a little protective of the character and I want them to feel like they're lost, so making the player character small, almost childlike, should help both of those feelings. It'll also subtly magnify the environmental dangers.

The prototype was quite bright and I want more of a 'Scooby Doo' feel, so I buy in some textures from **Gametextures.com** and remove all the ambient lighting, settling instead on wall-based torches that light each room while casting long, dark shadows. This immediately feels better, so I create a room where the player spawns in with a big particle effect and link it to another room with musical floor tiles that you have to hit in sequence (stolen from the *Billy Jean* music video, natch). Even the act of moving between rooms makes it feel like a proper game...

I think I'll aim for it to be a similar size to *Head Over Heels* – so roughly 150 rooms. My notepad has a list of the types of mechanics I'd like to build, so the job of creating rooms begins in earnest.

But there's another problem. My girlfriend hates the name *Arse Over Tit*. I've tried explaining that it's just a pun on *Head Over Heels*, but it's not washing, so I need to come up with something better. It seems apt to pick a Finnish name given that I've just moved here, so I fire up Google translator and start putting in words: Magic, Spell, Charm, Enchantment... Bang! There it is: *Lumo*.

What is Steam Greenlight?

Steam is a digital distribution service for games, applications and even movies that you're probably familiar with through its support for Linux via SteamOS. Originally created as a means for Valve to distribute its own games, it slowly opened up to third-party games after the release of *Rag Doll Kung Fu*. Rather than act as a gatekeeper, Valve sought to open up the process of accepting third-party titles to its community of users, and Greenlight was born. In principal the process is simple: games in the Greenlight process are publicly displayed and anyone who uses Steam is able to vote 'Yes' or 'No' to express their interest. In reality, what happens behind the scenes as Valve process this information and allows games to be distributed is a little murky, often causing controversy as games either avoid the process entirely, or languish in 'Greenlight hell' for years at a time.

Some of the neat gaming elements, such as blocks merging and floating away, needed to be carefully slotted into the level design so as to not break the immersive qualities of the game.

Linux software

Because Unity3D's editor didn't support Linux when the project started, most of *Lumo*'s development was done under Windows. But Free Software – running on Linux – played an important and crucial part. All of the audio, from the spot effects to music, passed through *Audacity*. *Krita* and *Blender* were ever-present. *ImageMagik* was used to bulk-process textures. The C# classes containing foreign language strings were generated by a Python script, which parsed a *Libre Office Calc* spreadsheet. The company website and blog is hosted on Ubuntu. *Git* was used for source control. Without FOSS *Lumo* simply couldn't exist, which is why my other machine is a Thinkpad running Linux Mint Cinnamon edition.

January – March, 2014

I've been having a bit of fun by sneaking in the odd reference to old developers. I doubt anyone will notice these but they make me chuckle. Some crates have "A.C. & G. Ltd" on them, as a nod to Ashby Computers & Graphics Ltd which, it was pointed out to me, became Ultimate Play The Game, before later turning into Rare. Carpets and curtains have borders that mimic the box art of early Ultimate games (*Knightlore* and *Mire Mare*) and there's a bit of graffiti on the wall saying "JR (John Ritman) & BD (Bernie Drummond) Woz Ere" above two little skulls. *Head Over Heels* made it into the game! I've also taken part in the first, unofficial, Twitter 'Googly Eye Challenge' after Rob Fearon suggested that my boxes weren't 'Rare' enough. They're now walking around with eyes that follow the player. Very cute.

Lighting is still in constant flux, but I'm liking the dark and spooky feel that the castle has, even if I do keep changing the colour palette. The ambient audio bed I've put in – consisting of little creaks and groans – seems to be working well and there's a distinct vibe forming. It's coming together.

So, I'm going to need to make the game public (rather than just my Twitter feed), and the quick way to do that is probably the Steam Greenlight process. I'm going to need to distribute via Steam anyway, but in order for this to happen I need to have some screenshots and a gameplay trailer. The latter is a problem. I'll need at least a minute, preferably two, of gameplay and it's going to need to show more than just running and jumping. It's time to make an instance of each of the game mechanics that I've scribbled down in my notepad and see if I can chain them together into something that looks like a game!

March – May, 2014

The first week of March is spent editing together the Greenlight video (**https://www.youtube.com/watch?v=7yQL8UP0TL0**). I have just over 30 rooms to choose from, mainly showing running and jumping, so there's plenty of footage. The inner DJ in me has taken over. I've ended up syncing everything to the music. Er, don't do that. If I'm honest, I don't have a plan on how to promote the Greenlight, I'll just see what happens.

I push the 'Go' button on 12 March and *Lumo* stays on the front page of the Steam Greenlight Community for ~5 days before dropping off, along with most of my traffic. I've posted to every related Facebook group that I can think of, along with G+, but, predictably, this isn't making much of a mark. I'm a bit worried that I'll be stuck in the process for a couple of years as I've heard some horror-stories...

And then there's a bit of luck. Rob Fearon championed the game in his monthly roundup and the day after **dome.fi** – a Finnish game and movie news site – picks it up. There's a noticeable spike in traffic and I make it to the Top 100.

I'm there for a week when Valve greenlights a new a batch games, so not being included is a blow and I have no idea when the next batch will be pushed through. I decide to send out some emails to gaming websites in the hope of getting some news pieces, but I hear absolutely nothing back. The traffic to the Greenlight page starts trending down sharply so I begin posting to Reddit and joining the #screenshotsaturday posts on Twitter. I continue this pattern, every Friday and Saturday, for a month, but I need a new screenshot of the game each week. It puts the pressure on being productive!

There's no obvious end in sight so I try another push with the gaming websites, but this time I include a mini CV of the AAA games I've made in the past and a story about how I've left the previous studio I was a co-founder of. Within an hour I hear back from Kotaku, which posts the news to its site the same day. The day after, Eurogamer picks it up, and then Rock Paper Shotgun. I'm elated and wait for the inevitable bump in traffic to the Greenlight page. Which never comes. Uh.

Steam's Greenlight process is a nerveracking rollercoaster of statistics and publicity opportunities.

The day after the RPS piece I wake up to people congratulating me over Twitter – for what I have no idea – until I check my email and see the email from Valve confirming that I've been Greenlit. Phew!

In the end *Lumo* was in the Greenlight process for 35 days. More than 30k people hit the page and there were 9k Yes votes. It's the first properly stressful part of the project, but it's done.

May–August 2014

I've told myself that I'm officially in 'Production'. That means doing some of the chores I've been putting off, like front-end, sound effects and actually settling some of the outstanding design questions. What powers will the player have? Will the game be centred around puzzles or skill-based? Should there be spells? How is the mouse best used?

And then I start getting some interesting emails. One of the console manufacturers is interested in *Lumo* appearing on its platform. They'd like to see my design docs and have me present my ideas. This is a bit of a problem, because – other than some notes scribbled on paper – there is no formal design doc. I've purposefully not written one. One of the things that's been so joyous about working alone is the lack of paperwork associated with professional game development. For the most part I just wake up and build whatever is in my head. I put a presentation together to start the conversation and hope they don't need too much, too quickly.

Up to now I've only been considering Lumo for PC, so assumed a mouse and keyboard control system (with the mouse being used to deploy spells, which I've yet to think up). Putting it on console would mean supporting joypad. This could work; in fact, it might actually suit what I'm doing, as the original 8-bit games only had one button. What if I stuck to the genre and made *Lumo* a one-button game?

By the start of August I have a very rough-and-ready build with a first pass at the game flow. I take this to Assembly (a demoscene/LAN party held yearly in Helsinki) to show it to a few people and it immediately becomes clear that no one has played an isometric game in years. Getting through my rooms is proving to be too much of a challenge. I need to seriously simplify the start of the game. Gah!

September–November 2014

I've entirely re-worked the flow of the game. There's now a section at the start that focuses on controls, followed by another 30 minutes gameplay where I'm slowly introducing most of the mechanics that'll be reused throughout the game. I'm not happy with how linear this has turned out, but I don't have many options, as anything else has proven to be overwhelming to new players.

I've also added an entirely new zone. My girlfriend and I stumbled on a lovely old church in the country so I snapped a few photos of its walls and made some textures out of them. I now have my own little church zone, full of cobwebs and spiders! I've also made great progress with the warp zone. Three of the mini-games are in and I've made some crazy little challenge rooms where everything is floating around the player. I'm really happy with how this part is shaping up (and the fact that there's a *Bubble Bobble* reference in there!)

Each section in the game has different textures and a different feeling, like the parts of a large castle.

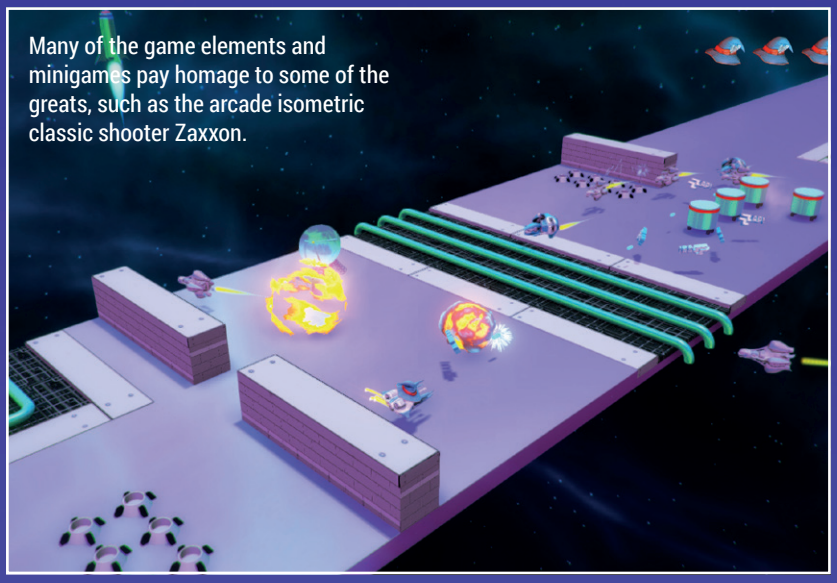
I've also got the Linux version working. There were a lot of shader issues that needed fixing up, so all the hard surface reflective shaders were removed and the water replaced. But it's rock solid, and performs every so slightly faster than the Mac version on the same hardware. Happy with that.

December 2014

I've changed the lighting, again. I've now got two 'suns' in the world, one for each of the back walls. These are different tones (one warm, one cold) so everything's starting to pop a little more. I'm also colour-grading the final image. This is a lovely little trick that takes the colour of the pixel intended for the screen and pushes it through a look-up table. The look-up table enables you to bake-in adjustments – contrast, saturation, brightness, tone etc – so gives you fine control over the final image, without having to do work to the individual textures. I remake the Greenlight video (**https://www.youtube.com/watch?v=M7vVXTeCepA**) at 60fps to show this off.

January–March 2015

I've hit a nice rhythm with what I'm building now – ideas are coming thick and fast – but this is the busy part of the year for teaching. I'm only getting a couple of days work done in between the commuting. I still have a



Many of the game elements and minigames pay homage to some of the greats, such as the arcade isometric classic shooter Zaxxon.

lot of conversations happening in the background about how to release the game, which is eating up time. Progress is slow, I'm low on cash and starting to sweat a little.

The book model is done, I can animate pages, and I can render stuff to each of the pages, but this was far more difficult than I expected. And it's had some nasty side-effects, the biggest of which is pausing the game. Up to now I was just freezing time, so everything in the world was locked, but I can't do that and have animations and particle effects play reliably as the book moves through the world, so in-game objects now have to detect they're paused. I'll be chasing little bugs around this forever, I know it.

I've got a Vita version working, albeit slowly, which shows some potential, but I don't really have time to optimise it. I drop this for now.

April 2015

Unity – the game engine I'm using – has bounced up to version 5. In some senses this is an improvement. Up to now I've had one version of Unity for PC and different versions for every console, each of which had its own bugs. With v5 everything should be unified in one editor, with plugins to support console.

But there are other changes under the hood. The code API has changed. Fortunately none of my code was affected – I tend not to be fancy and was already doing the things that the API change was brought in to solve – but the same can't be said for code I was using from the Asset Store. I decide to re-write all of this. I also bin the stuff used to create the UI in order to use Unity's new UI tools. This is a complete re-write of the front-end, in-game and mini-game UIs. Ugh.

If that wasn't nasty enough, the lighting model has changed. I'm reliably informed that the legacy deferred pass won't be optimised on console, so I have to use the new GI/deferred lighting model. This isn't a small change. Every single material and shader in the game needs to be re-done. Every room needs to be re-lit. I no longer have the old fog model. Kill me.

May–August 2015

This is, by far, the most productive I've been on the game. The missing mini-games are written, all the cut scenes are done. I've made well over 100 rooms during the summer break. The game moves through the tutorial section, before slowly opening up into the final hub zone, and I think it's starting to feel good. The re-factoring for Unity 5 has stabilised and the three main skus are all running well and in sync. I've no major platform-specific bugs.

Everything I've thought up has worked first time and I feel like I'm working like a well oiled machine. More importantly, I'm finally making the stuff I envisaged at the start of the project.

I've also signed a publishing deal. This is a much-needed injection of cash and support, so for once I'm not stressing about that side of things.

September 2015

The publisher is taking *Lumo* to the EGX Rezzed show in London. This will be the first time the game's been in other people's hands since Assembly, a year ago. I'm nervous as all hell. I also need to make a specific build for the show and a new promotional video (**www.youtube.com/watch?v=fdK-HlHavM4**).

Fortunately the reaction from the show is amazing. Several people are calling it the game of the show, which is extremely flattering. Eurogamer do a video on it and suddenly *Lumo*'s in the spotlight. I'm quietly recovering from the palpitations I've been having all month.

October–December, 2015

I've agreed to hit content complete by January so that Just Add Water Developments can begin on the console ports. I've got one section of the game left

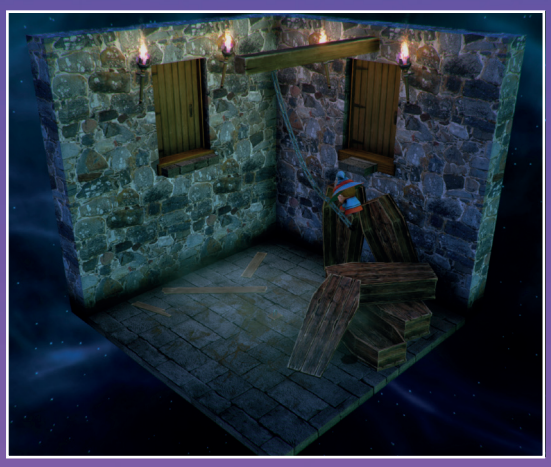
to do, about 30 rooms, then Steam integration, save game, map, stuff like that. Seems fine. Of course, it doesn't work out like that.

Most of November is knocked out with travel to the UK for meetings, bits of PR and teaching. I get all the rooms done, just, but I need to balance the game and do all the integration stuff I've been putting off. December becomes an almighty crunch to get everything done. I'm working every waking minute and starting to stress a little. The situation's not helped by constant bugs in the Unity engine. Animations are breaking, the particle systems are not firing, or flicking randomly about the world. UI rendering picks up random garbage occasionally. Reflections are dropping the frame rate to unacceptable levels and I have a crash on Linux that I can't debug. I roll back to an earlier Unity version and things are stable enough to get to content complete, but I've just had a glimpse of life over the next four months.

January–March 2016

This should just be a bugfixing phase, but it doesn't work out like that. I find out there's a problem with Unity on Xbox One that requires me to pull out all the audio using Unity's in-built system and integrate it into FMOD. There are thousands of audio instances spread around the game and after December's antics I'm in full 'Hulk Smash' mode.

I'm also struggling to find a version of Unity that's stable across Linux, Mac and PC. Each has different bugs and I have crashes on Mac and Linux. In the end I run with three different patch releases of Unity, one for each platform, and I do hacks on my side to get over individual platform bugs. Some of it's filth. Because of this I also have to move off Unity 5.2.x and go to 5.3.x. Another API change has been made, which affects a fair amount of my code. All the particles need fixing, again, and the scene loading is different. This introduces a weird race condition into the start-up of the game that turns out to be a subtle change to the serialisation order of components. I hack around this and I'm back to where I started.



The subtle lighting in the game caused a real problem when a Unity update changed everything as the game neared completion.

Even *Lumo*'s cover design is reminiscent of computer game adverts from the 1980s.

It's not been a good upgrade, but things get worse. The specular lighting value in the rendering engine has changed. Every single specular material, in-game, is broken. I could actually cry at this point.

I fight through the process of re-doing the lighting and checking the materials in every room of the game. It's a rush job, because I have artists jumping up and down at Just Add Water who need the scenes for the console builds. I get things close, and throw the build over the fence. Looking back I didn't do a great job at this – some of the rooms don't look anywhere near as good as in previous builds and the intro scene ends up being flat and lacking contrast. None of the metals look good. I'm probably the only person that'll notice, but it's annoying nonetheless.

I head to San Francisco for a PR trip, tired and worried about how the US press will take the game...

April–June 2016

We're due out on the 24 April but we're not going to hit the date. Unity bugs blocked one of the console versions right from the start and we're stuck in cert with another. For everyone's sanity the date gets pushed back a month and I'm in limbo.

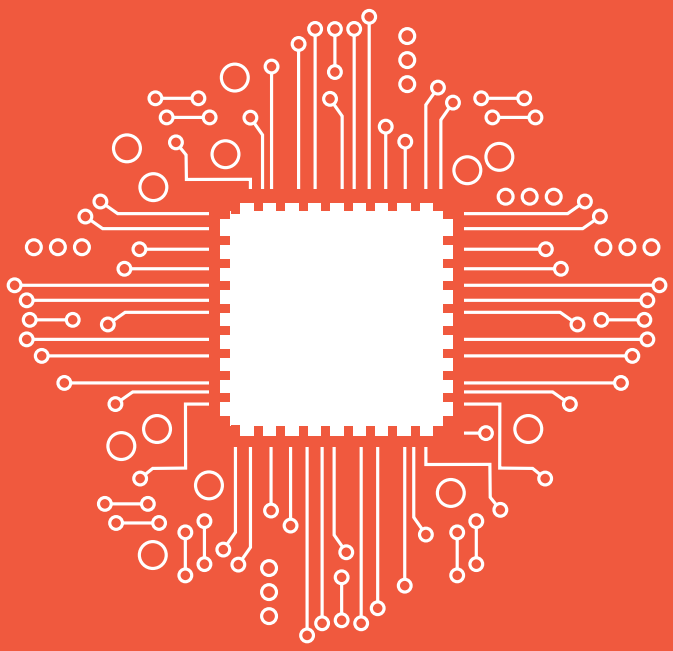
In the end we hit the 24 May date with four of the skus. Xbox one and Vita follow behind. I receive an email from Eurogamer to let me know that they loved the game, and finally I get a decent night's sleep.

The rest of the month is a blur of interviews, podcasts, 4am Twitch streams and fixing some small bugs I notice as people are playing. I also start on the Director's Commentary series for YouTube.

It's been a rollercoaster getting to this point, but I've done what I set out to achieve. *Lumo*'s a niche game, admittedly, but watching the first speed-run of the game on YouTube and hearing that people have already rinsed it for the achievements is a fantastic feeling. Turns out there were quite a few people – like me – who loved those old isometric games. I'm hoping someone else picks it up from here and makes some more.

Rendering

As GPUs have evolved, so has the way we program them. In the early days hardware was 'fixed function' – geometry and materials were passed through the pipeline and processed in a pre-defined way – but that restriction was quickly lifted so programmers were able to write bespoke code (shaders) to define these operations themselves. This allowed for differing ways to handle light in the world. In the fixed-function days lighting was inflexible, but with the advent of shaders it was possible to process all the geometry before calculating the lighting (deferred), reducing the cost and increasing the number of lights you could have in a scene (with some caveats). Due to the quest for more photo-realism, many game engines have moved beyond rendering how direct light affects objects to include indirect lighting, or 'light bounces', as well. This is called 'global illumination', but because of the constraints of having to render a scene in real time, this is often a cheat and may only include 'ambient' light, often taken from something like the sky that surrounds the world.



OLD HARDWARE NEW GNU/LINUX

Andrew Conway never throws anything away – not when he can install Linux on it!

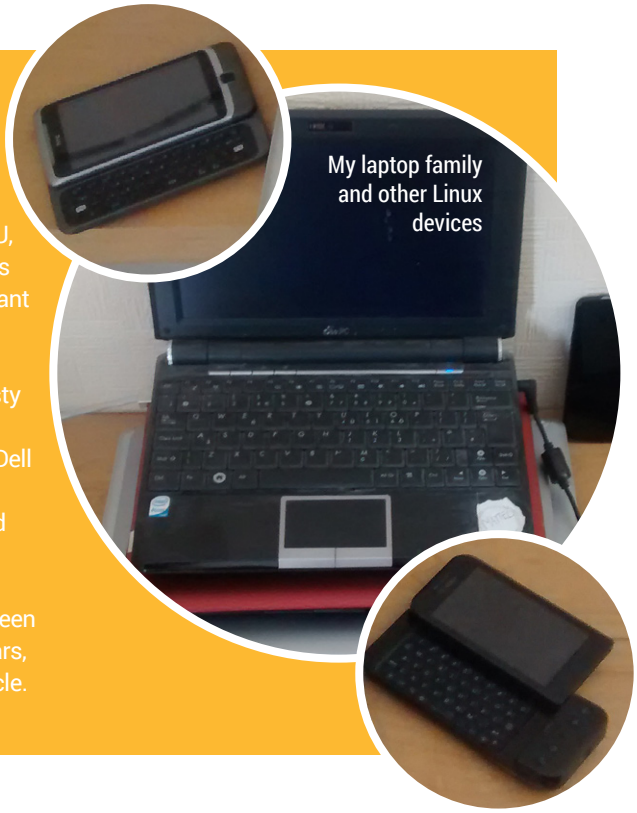
What's the oldest bit of hardware you still have in active service? A few weeks ago, a laptop I'd given to a relative came back to me. But this wasn't any old laptop: I'd bought it over a decade ago and it was the very first laptop on which I installed Linux myself. Call me sentimental, but it was like an old friend returning from overseas. Sadly, it was sick. I assumed some hardware had finally failed and I would have to say farewell. But no, it turned out that the old version of *Firefox* shipped with the ancient version of Linux Mint wasn't able to cope with Gmail. I installed a more recent distro on it and it was soon back at my relative's house doing its light web and Gmail duties more capably than ever before.

Us Linux users are loath to abandon older hardware because we know that the right distro can breathe life back into it. Here's the story of a few devices over the last decade and their journeys from Windows through various Linux distros.

MEET THE FAMILY

Here's the family photo. Apologies for it being somewhat dusty but they're having to pose down the back of the TV because old Auntie Eee PC cannot leave her wired connection as she's gone a bit deaf in the old Wi-Fi department and is busy doing essential server jobs. The handwritten Mattel label was affixed by a jealous MacBook user. Grandpa EasyNote is the oldest on show and despite being 11 years old is in rude health except for his dead battery and a mysterious scorch mark near his LEDs.

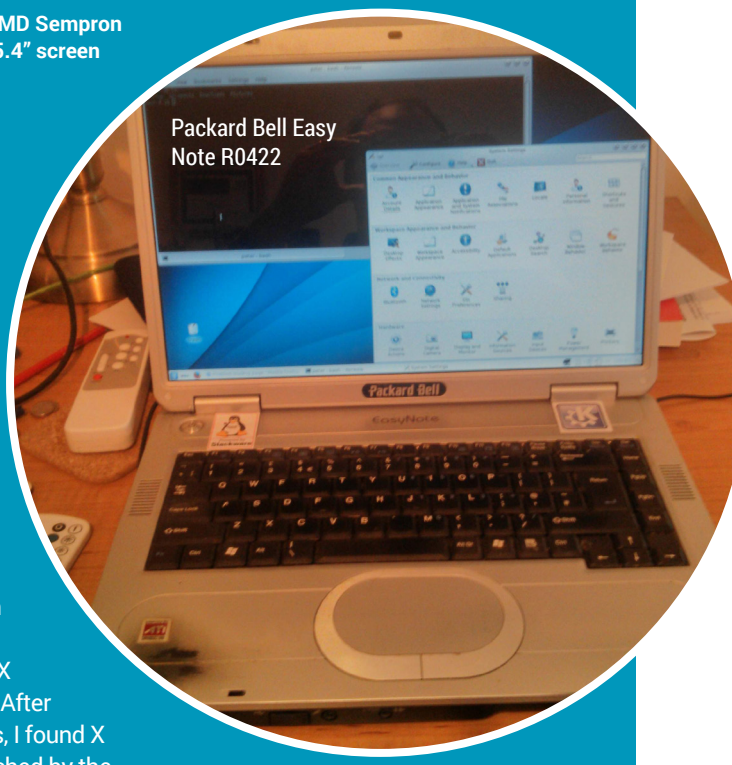
EasyNote's granddaughter, Dot U, also of the clan Packard Bell, has benefited from a battery transplant some years ago. Although her hardware is in good nick, she's a little flimsy and developed a nasty crack at the laptop hinge. The whippersnapper, young master Dell XPS 13, is fast, elegant and well built. Although he had a troubled childhood which involved major motherboard surgery and some temporary kernel hackery, he's been a sturdy workhorse for three years, and is helping me write this article.



My laptop family and other Linux devices

PACKARD BELL EASYNOTE

Shipped OS: Windows XP; My OS: Slackware 11,12&14 , Mint 11 ; AMD Sempron 3100+ 800GHz; 45GB HDD; 512MB upgraded to 1.5GB RAM; 3kg; 15.4" screen



Packard Bell Easy Note R0422

In 2005 I spilled a mug of coffee over my laptop and it died. I had urgent work to finish, so I rushed out to the nearest high street store and bought the cheapest new laptop I could find: a Packard Bell EasyNote for £299. It was only meant to tide me over for a few days, and then become the office spare. But to my surprise it had a good screen, sturdy keyboard and ran smoothly enough. Under Windows XP, I used *Outlook* for email, *NetBeans* for Java coding, *Microsoft Office* for documents and *Internet Explorer* for the web.

Enter, Linux!

Then over lunch one day soon after, a friend of mine recommended I read *The Cathedral and the Bazaar* by Eric S Raymond. I did, and after reading further afield I found that Linux was still alive and kicking. You see, I'd used Linux on my desktop PC from 1995 to 1999 and for scientific data analysis in the following years, but by 2004 I'd ended up in Microsoft-land because I'd left academia and was working in a more traditional office environment. Despite my history of Linux use I was quite ignorant about GNU/Linux's free software underpinnings.

My first step was to replace *Outlook* with *Thunderbird* and *Internet Explorer* with *Firefox*. Next up was replacing *Microsoft Office* with *OpenOffice* (no *LibreOffice* back then). After a bit of reading around I decided to switch to Linux – Debian – but I got rather confused by the fact there were various different Debian ISOs to download. The prospect of installing an operating system was daunting enough for me without the added confusion of understanding what the different distro editions were for. The matter was settled when I mentioned my conundrum to my Dad, who reached into a pile of DVDs and handed me one that had Slackware 11 written on it. I took the DVD home, stuck it in my desktop machine and, after reading some instructions, managed to get it installed, dual-booting alongside Windows 2000.

Installing Linux on the EasyNote was much less straightforward. The first issue was that X windows wouldn't start. After some random fumblings, I found X windows worked if launched by the root user, and some web research told me the problem could be fixed with a permissions option in a file called **xorg.conf**. This was the first of many occasions where I'd find myself rummaging around inside **xorg.conf**'s innards: sometimes just to get X to work, other times to exploit X's flexibility in getting unusual display configurations working. These days **xorg.conf** gets automatically generated on the fly and I've not had to edit it in years.

> After a bit of reading around I decided to switch to Linux –Debian – but I got confused by the various Debian ISOs

But the battle with the X server was minor compared to the beast that was the Wi-Fi driver. The EasyNote's Ralink RT2500 chipset was still quite new back then. The first issue was that Slackware 11's 2.4 kernel didn't have a driver for it, but there was an option to switch to the 2.6 kernel. Doing this provided a driver, but still no network device. After much searching I found some source code for an updated kernel module released by the manufacturer. When perusing the various files that came with it, I found some very handy notes made by a chap called Alan Flavel. By some great coincidence, Alan shared an office with my Dad many years before at Glasgow University, but the touching aspect to receiving his virtual help was that he had died the year before. Thanks Alan.

Once I'd got Linux set up, it was simply a matter of importing emails and bookmarks from *Thunderbird* and *Firefox* on windows over to Linux. Also, as *NetBeans* runs on Java, it was easy to get my Java development environment up and running. *OpenOffice* worked fine too, and so I switched to working under Linux full time and I've never looked back.

FROM NETBOOK TO SERVER

Shipped OS: Xandros; My OS: Slackware 13, Tiny Core, CentOS 6 ; Intel Atom 1.6GHz, single core; 160GB HDD; 1GB DDR2 RAM; 1.45kg; 10" screen; £349

The main problem with the EasyNote was its weight. At over 3kg it became a literal pain in the upper arse (lower back) especially as I found myself travelling a lot more for work. For this reason, and because I now understood how to get the most out of fairly modest hardware with Linux, I bought a netbook: the ASUS Eee PC 1000H. When on my desk it had a separate keyboard, mouse and an external monitor. It got a lot of use on the Glasgow to London train and with battery upgrade it was easily capable of lasting a five-hour journey, but working on a 10-inch screen presented problems even with KDE in netbook mode.

I experimented with a tiling window manager – dwm with dmenu – as having each window using the whole screen made a lot of sense. Alas, although *dwm* was stunningly fast and worked well for most things, *NetBeans* refused to work on it, so I went back to KDE. KDE 4 was still a little buggy at that time and it let me down badly one day when the KDE taskbar crashed and froze the clock, which made me late for picking up my six-year-old son from school. A message to developers: software bugs can lead to tears!

Time to upgrade

The Eee PC solution served me well for some time but eventually the Intel Atom single-core processor and 1GB of RAM proved too limited for certain tasks (in particular virtualisation). I replaced the Eee PC with the Packard Bell Dot U for daily use and turned the Eee PC into a server.

The EeePC draws about 12W, which is not much above a Raspberry Pi with a separately powered USB hard drive

I installed a minimal CentOS 6 on the Eee PC, diverted incoming traffic through my home router to it on port 80 and some others (I don't use 22 for SSH, nor 443 for HTTPS) and got *Apache* running. Next, with the help of some Dynamic IP services (**dyn.com** and later **noip.com**) I set up the DNS for various domains that I owned so that their websites could run on the Eee PC.

The EeePC draws about 12W, which is not much above a Raspberry Pi server with a separately powered external USB hard drive attached. The 160GB internal drive is fine for its server duties, but I noticed it suffered from hard drive clicking. This was caused by its head parking every eight seconds or so, and as the design lifetime is typically for 300,000 parks, this means that if run 24/7, it would reach its design limit in a month or so. The solution is to issue this command:

```
hdparm -B 255 /dev/sda
```

This tells the first hard drive (**sda**) to run without any power saving optimisations and you can ensure it runs on every boot by placing it in the file **/etc/rc.local**.

Encouraged by this success I installed *tt-rss* (tiny tiny RSS) which manages my RSS feeds, and *OwnCloud*, which I use mainly to manage files. *OwnCloud* worked well at first but in time problems emerged with the desktop sync client. It'd report the server giving mysterious 500 HTTP errors and conflicts were flagged for files which definitely were not conflicted, ie the local and server files were identical. This was annoying but seemed to be harmless and after checking carefully a few times I got into the habit of deleting the pointless conflict files.

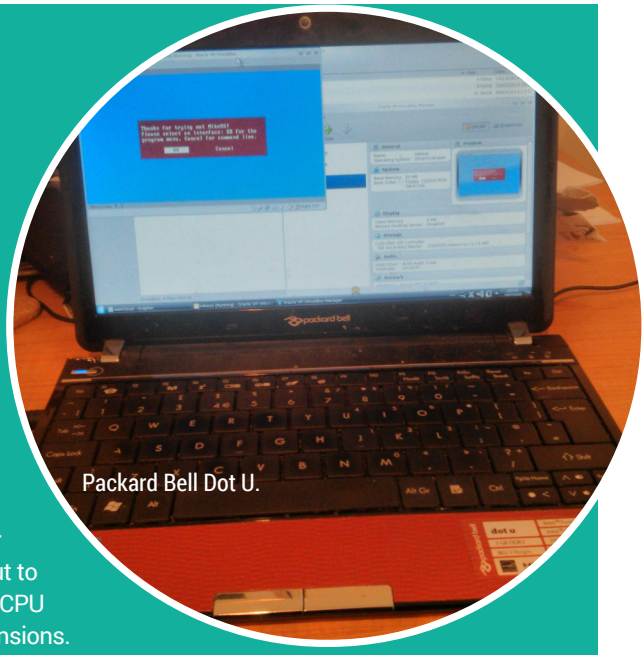
But this was a bad idea. It finally came to a head when I lost a chunk of a book I was writing because the *OwnCloud* sync client replaced a freshly edited text file with one from the week before. What's worse is that I'd deleted the conflict file with the **rm** command and the *OwnCloud* directory only got included in my weekly, not daily backups. A week's work gone. The problem turned out to be that I'd set *SQLite* as the back-end database rather than *MySQL*. The *OwnCloud* team recommend you don't use *SQLite* on a production server and I'd just found out why. It's probably OK for a small set of files, but mine had grown to thousands of files and GB of data over the years. Fortunately the *OwnCloud* **occ** tool made short work of the *SQLite* to *MySQL* conversion, thus solving the problem.

Asus EeePC 1000H.

THE YOUNGSTERS

PACKARD BELL DOT U Shipped OS: Windows 7; My OS: Slackware 13; Intel US5400 1.2GHz, dual core; 250GB HDD; 3GB DDR3 RAM; 11.9" screen; 1.3kg; £450

DELL XPS 13 DEVELOPER EDITION Shipped OS: Ubuntu 12.04LTS; My OS: Slackware 14.1; Intel i7, 2GHz, dual core, 4 threads; 256GB SSD; 8GB DDR3L RAM; 13.3" screen; 1.36kg; £1,100


Packard Bell Dot U.

The Eee PC's replacement, the Packard Bell Dot U, didn't feel that much faster, but the slightly larger screen and lesser weight made it more convenient for use on the move. The extra 2GB of RAM over the netbook was also welcome, especially for running virtual machines, but to my dismay the Intel U5400 CPU lacked virtualisation extensions. This made a big difference in some cases; for example, *Qemu* fell back on emulation and so ran like a slug on a marijuana plant, but thankfully *VirtualBox* performance wasn't too bad (for reasons I never quite understood!). Although I didn't choose this laptop for its Linux compatibility, it was so easy to install Slackware that it was almost boring. Everything worked out the box.


Dell XPS 13 developer edition (2013).

My current laptop is the Dell XPS 13 Developer Edition, which came pre-installed with Ubuntu 12.04LTS but currently runs Slackware 14.1, and will soon go to 14.2. I can safely say that after three years of daily use – once a few teething problems were resolved (see my articles on Slackware in LV006 and Kernel Parameters in LV009) – the XPS 13 is the best laptop I've ever owned. It's also the most expensive!

EMORTALITY

What will bring these devices to the end of their lives? A failed hard drive is probably most likely, but this is not terminal, as hard drives are reasonably easy to replace, even in laptops. In fact, I'm expecting this for my Eee PC, and my plan is to replace its aging spinning hard drive with a fast SSD one. The bigger worry is failure of bearings inside the fan – if it stops spinning then the overheated CPU may well suffer terminal damage.

But there's another issue. CentOS 6 will stop getting full updates by June 2017, and maintenance updates will end in November 2020. Well before that time, the third-party repositories such as EPEL might not provide recent enough versions of PHP for *OwnCloud* and other key software. There will be no upgrading the Eee PC to CentOS 7, because it is only available for 64-bit architectures and the Eee PC's Intel Atom is 32-bit.

This mortal coil

No server lasts forever of course, so at some point in the next two years I expect I will either press my 64-bit Packard Bell Dot U into service as a server, following an SSD drive transplant, or else use a Raspberry Pi 3 running Raspbian with an external USB SSD drive. My preference though is to keep my old hardware in use for as along as possible with the proviso that it is not too power-hungry relative to more modern technology. I'd like to say it's because I'm conscious that we live on a planet with limited resources, which is true, but if I'm honest it's also because I've a certain, sentimental attachment to my aging devices especially if Linux can make them useful.

FAQ

OSVR

The new standard in Virtual Reality could be open source.

MICHEL LOUBET JAMBERT

Q So what does OSVR stand for and who's behind it?

A Really? Open Source Virtual Reality... Seemed kind of obvious, though I guess it could have been Operating System Virtual Reality, but that doesn't make much sense. As for who's behind it, it's an open project with a bunch of industry backers, ranging from big-name video game industry companies to lesser known manufacturers, as well as a few academic institutions interested in the educational and technological aspects of VR over its gaming uses. If you're familiar with game developers, then some well known names are Ubisoft, Techland and Gearbox, the latter two of which have games out on Linux already.

Q That's great, but isn't OpenVR already a thing?

A True. OpenVR is the VR ecosystem developed by Valve and HTC for their Vive VR headset and controllers. However, there are a few caveats, the most striking of which is that it isn't available on Linux yet, despite it having being designed to complement the Linux-powered Steam Machines also developed by Valve. Support is expected to arrive at some point, but who knows when. The name is also somewhat misinterpretable since, though the source code is available to look at, gratis, modifiable and redistributable, it is more restrictive in the sense that it must retain the Valve copyright.

Q Fair enough, but with Oculus Rift, OpenVR and a few others, isn't VR already fragmented enough considering it hasn't really reached the mass market in any significant way yet?

A That's a good point, but then this often happens when a new technology comes out and a standard gets established. Remember VHS vs Betamax or Blu-ray vs HD DVD? This is kind of the same thing, and while there will be a number of hardware manufacturers, the theory goes that "market logic" will determine which standard drives that hardware. History shows though that, it's not always the best one that wins, nor is it logic that determines the winner in many cases.

Some advances are being made on the fragmentation side of things though, and Valve has already announced that it will be collaborating with OSVR. This could mean any number of things so there's not much point in being too speculative, but it does show that Valve isn't intent on putting up a fight and possibly willing to let the best standard win. However, if I had to put money on it, I would say that some sort of open (or semi-open) standard will come out on top, considering that on the development and OEM side of things, people will want to look at the source code and make changes to suit their software or hardware, based on how successful Android has been in the mobile space.

> It's an open project with a bunch of backers, ranging from big-name video game industry companies to lesser known manufacturers

Q So what sets this technology apart from the rest?

A Well, as I alluded to before, there's the Linux support and the whole free software licensing thing – more specifically, OSVR uses the Apache Licence. There's also the hardware, with

each manufacturer having developed their own standard to go with it, but OSVR leaving the door open for anyone to grab it and sell their own hardware. Like Valve's policy of sending advances of prospective sales to developers, OSVR also has its own financial incentive in the form of a $5 million fund which developers can apply for if choosing the VR ecosystem, which could be more attractive since it isn't necessarily based on sales.

The Razer HDK2 uses the OSVR standard . (photo: OSVR).

Q What's the hardware like for OSVR then?

A The current offering is a lot cheaper than the Oculus Rift and the HTC Vive, that's for sure. The second generation HDK2 (Hacker Development Kit 2) from Razer sells for around $399, compared to $499 for the Rift and $799 for the Vive, so it's certainly the more accessible option. Most reviews do claim it to be not quite up to standards with the other two in terms of visuals, but then VR as a whole is far from perfect right now, so that should be taken with a grain of salt.

In specific terms, the HDK2 has a 2160x1200 OLED display refreshing at 90hz, an IR camera for position tracking and decent user adjustment options. On the controller side of things, the HDK2 doesn't have its own, unlike the Vive. However, it has the advantage of supporting a multitude of different devices, ranging from XBox controllers to WiiMotes and competing devices, showing off some of the advantages of an open model and leaving things open for OEMs to create their own peripherals. All things considered, it might well be the most appealing device out there in terms of cost-performance ratio, and the only device usable with Linux. However, given the small amount of VR titles available as a whole and even less using OSVR, none of these devices are that appealing right now to those who aren't either gaming enthusiasts, tinkerers or developers looking to take their first steps into VR.

Q So where do I sign up to get developing with this thing?

A A good start would be to order a set first – it's called the Hacker Development Kit for a reason. You should also head over to **osvr.github.io** for some resources and documentation to get you started as well as guides on how to use OSVR on some different game engines. Speaking of which, it would also be a good idea to get yourself a game engine installed to develop on. Currently supported engines include the Unreal Engine, Unity, CryEngine and Blender, so there's already a good mix of gratis and libre Linux-supported engines with which to build games and other things, and even create commercial-grade games considering these are among the most widely-used out there. If you're new to game development, something like Unity would be a good bet, since it's user-friendly and there are plenty of tutorials and free assets available online, though Blender Game Engine is a great option if you want to stick with FOSS only.

Q How do you see the chances of success for OSVR as a virtual reality standard?

A It's too early to say with certainty. The cooperation with Valve is a good start since, although VR could have many potential applications, gaming is likely to be the area in which the teething stages of the technology are carried out, considering there's a market for it already, and having the standard supported by the biggest vendor of PC games in one way or another is a very good start.

Being an open standard also has a lot of benefits, not just from an ideological point of view, but more that it keeps doors open for third parties to make their own hardware and adapt it to their needs, as we've seen with other cases. With OSVR, there's potential for console manufacturers to use the standard with their own headsets rather than developing their own from scratch, which is a lot more likely than some sort of partnership with the other big companies behind the other standards, many of which are competitors. This logic also applies to other uses for VR, such as research or more adventurous companies looking to create new forms of entertainment. Likewise, the cross-platform nature of an open standard such as this one is a huge advantage in gaming specifically, since developers want to sell as many copies of a game as possible and being locked out of certain platforms, or having to incur extra costs of porting to them, is a huge drawback.

Then again, with that logic, DirectX shouldn't really be as pervasive as it is, which goes back to these things not always being defined by logic. One thing that is certain though is that it will come down to a mix of adoption by both consumers and developers, since people will want the platform with the most games and developers will want the platform with the biggest user base. Success is also largely tied to the success of VR as a whole, since if it turns out to be niche, there's only a small pie for the OSVR standard to take a slice of.

On the flipside, if the VR market turns out to be larger, then an open project which doesn't require vast amounts of money being pumped into it has its clear advantages. Check back in a year or two for more clarity, but OSVR has a fighting chance in the meantime.

> Maintaining lots of Arch Linux machines from afar meant that my Christmases and Easters and family holidays were basically me doing tech support for everybody in the family

MARTIN WIMPRESS

The man who's everyone's Mate

As well as being one quarter of the Ubuntu UK podcast, Martin Wimpress is also responsible for the youngest official Ubuntu flavour – Ubuntu Mate. It may be new, but it's already made a name for itself as a blend of new technologies with a classic interface. It's also the first of the desktop Ubuntus to run on the Raspberry Pi, thanks in large part to the massive efforts that Martin puts into it. We caught up with Martin to find out what to expect in coming versions of the distro.

Can you tell us a little bit about how Ubuntu Mate got started?

Martin Wimpress: Well, the genesis of the project is really that I got involved in Mate itself. Very briefly, I was using Ubuntu but wasn't happy with the way Unity was going in the very early days, so I switched over to Arch. I was using Gnome 3 and I switched my wife's laptop over to Arch, also using Gnome 3 – this is about the same time that she got her first Android phone, and she did everything on the phone. A couple of years go by and she really hasn't used her laptop at all, then one evening she decides that it's the time to sort out baby photos and tries to use Gnome 3 – stern words were exchanged as she wasn't familiar with this new fangled thing. Very hastily, I removed Gnome 3 from her Arch laptop and put Mate on (I wasn't involved in the project at all at that time). This got back what she was familiar with (because she'd been using Ubuntu since 6.06) and wedded bliss was restored. She could organise the photos again because it was all as she expected.

As I put Mate on, I noticed that there were some bugs and rough edges, specifically around the packaging for Arch. I was thinking, "My wife's happy with this, I need to fix these things so I don't have the next

Like us, Martin's a podcaster – tune in to ubuntupodcast.org to hear his measured, calming tones.

problem and the next problem..." I started contributing fixes to Mate and working on packaging it for Arch. That's how I got involved with Arch. Roll on a couple of years, and most of the family (who I had moved over to Ubuntu in the 6.06 era) are now running Arch with Mate on their laptops. That wasn't working particularly well for them because Arch Linux is an attention whore and maintaining lots of Arch Linux machines from afar meant that my Christmases and Easters and family holidays and get-togethers were basically me doing tech support for everybody in the family. They weren't happy with that and it wasn't going particularly well for me.

Then I did an interview about Mate on the Ubuntu Podcast long before I was involved in the Ubuntu Podcast. In this interview I pointed out that Mate was completely bust in Ubuntu. Popey [Alan Pope, podcaster and all-round nice guy] was just outraged that this was the case – this was off the air. A couple of weeks later he sent me a link to an ISO image that he'd made that was Ubuntu 14.04 with the Mate desktop stuck on top. It was very rough, but I thought, hmm OK, this could solve the problem – it's a way that I could give my family back that familiar Ubuntu platform with that familiar environment. The genesis of the project was Popey making that initial draft.

Popey and I then took a day off work – we went over to Popey's house and we did a one-day sprint on how this was going to work. We basically fleshed out the goals of the project and set the wheels in motion on that day. The purpose was to recreate that experience that my family were familiar with, because they're not technical – they don't want to re-learn how to use their computers, they just want to pick it up and do what they want to do, put it down and forget about it – they're not interested in learning anything new. So, that was the origin – making what they had before a thing again. It sort of snowballed from there.

When you say that it sort of snowballed, was there a lot of pressure from users to keep developing it?

MW: When I started on it, I thought "I need people to test it so I don't give something to my family that's completely broken. If I give it to people then I'll get some feedback about it." I did that just on Google+ initially and it spread like wildfire. Word got around and within a few weeks I was very aware that there was a substantial number of people who were interested in this thing. That's when I stepped thought that if there's potential for it to be popular, I'm going to go all in – I'd only got my pride to lose.

I started to make alpha and beta images and the download numbers were constantly growing. There was definitely a groundswell of popular opinion, and it's just grown from that.

> The core team that contribute regularly is about 7 or 8 people

How big's the team of Ubuntu Mate?

MW: Well, the core team that contribute regularly is about seven or eight people. The lion's share of the packaging work is actually done in Debian, so what makes Ubuntu Mate, nine tenths of that actually happened in Debian directly and it just syncs into Ubuntu because that's the relationship we have. I do most of my work in Debian – the only work that happens in Ubuntu is branding, artwork, and Ubuntu Mate Welcome, which is our centrepiece.

Within the Debian team, there are three people (myself and two others),

then a lot of the work is driven through the Mate desktop team itself – there's a core team of about five contributors.

Ubuntu Mate has several crowdfunding platforms and we try to push as much of that money as possible back into development projects for Mate desktop or Ubuntu. We draw in money on a monthly basis and I pay all the hosting and then I work with the developers and look at the projects that we're interested in developing, and then pay them to work on specific tasks in a given month. We pay as close as we can to the market rate for development time for them to work on things.

The Mate desktop team has about five or six people, then we have a pool of people specifically on Ubuntu Mate itself. They are not involved in Mate upstream, they're not involved in Debian, they're working specifically on Ubuntu Mate projects. That's two people in addition to myself. There's a cast of thousands that do artwork and they look after the forums, and just a ton of stuff that I can't do. I can't do anything artistic or creative at all, so I lean on the community for all that stuff.

Hands up who else does *pro bono* tech support for their family when they'd rather be chilling out?

LV You mentioned that you created Ubuntu Mate because you wanted a distro that your family could use because it was familiar and unchanging. Is this still how you see the future of Ubuntu Mate?

MW: Technologies underneath Ubuntu Mate are changing rapidly and we're moving with those changes. Ubuntu Mate 16.10 is built entirely against *GTK 3*. That's an ambition that we've been pursuing for a few years now, and we're the first distro to say, "right, we're going to jump". We've just had 16.04 – a nice stable LTS [Long Term Support] – now's the time to break all the toys and start again, which is what we've done.

We've invested about €3,000 since 16.04 in development projects to drag the *GTK 3* support up to snuff. We've got one outstanding issue at the moment that we've got somebody committed to do the work and the money set aside for. But the move to *GTK 3* is not a change in the user interface or the user experience, it's just using *GTK 3* to recreate that user interface that everyone's familiar with. The changes, the new features, are very small in Mate. Mostly it's to do with bugfixes and tweaks and very small additions, not a wholesale overhaul. The UI is buttoned down, it's defined, it's as it was documented by the Gnome team all those years ago – we're not changing that. What I do in Ubuntu Mate is try and build some tools around that to make it a bit more modern and make some conveniences that a modern desktop should have, but that's not changing the fundamental base.

LV Is this an ongoing commitment to not change the way Mate works – how it is now is how it will be in 10 years time?

MW: Yes. How it is now is how it will always be. It's a very unglamorous project to be involved in because if your user interface is your shop window, that's unchanging. What's changing is keeping up with the underlying technology, so along the way we've adopted support for thing for things like *GSettings* and tossed out *GConf*, and we've adopted support for *systemd* you know – these are all things that you need to do to make a desktop work in a modern Linux operating system, but in terms of what the user sees and gets, it hasn't changes. We've removed a couple of minor things and we've added a couple of bits and pieces, but over all, it's just as it used to be, and that's how it continues to go.

LV I believe that Ubuntu Mate was the first desktop version of Ubuntu to work on the Raspberry Pi.

MW: Yes, it was.

LV Was that something at you pushed?

MW: Way way back, I made a tool to build an Arch Linux installation that works on the Raspberry Pi – this is the original Pi with 512MB of RAM. I never quite finished that project and I always wanted to make a desktop for the Raspberry Pi. When the Pi 2 came out, Popey said, "you should totally make a version that runs on the Raspberry Pi", and I thought, I'll do that when I find the time. Someone in the community, Rohith Madhavan, made the initial port and it was eight tenths of the way there. Because so much of the work had been

done, I decided to pick it up and run with it, and from that point onwards I've maintained the Raspberry Pi build.

We had a hinkey version for 14.10, but the first proper version was 15.04, and I've really focused on that – it's our most popular version now [about half of all Ubuntu Mate downloads are for the Raspberry Pi]. Most people who are running Ubuntu Mate are running it on a Pi 2 or a Pi 3. We've also made root filesystem available for ARMv7 so you'll find it when you ask for ODroid Ubuntu, it actually comes with a sort of Ubuntu Mate now, and the same on the Banana Pi, Pine 64 and so and so on.

It's kind of become this ubiquitous Ubuntu thingy that you can run on all of the ARM devices. As a spin-off from that, I've become involved with the Raspberry Pi Foundation and I was an insider for the Pi 3. I got a prototype, and I was able to do some work so that on launch day for the Pi 3 we had a version of Ubuntu Mate that supported all of the hardware features that the Pi 3 had to offer. That's kind of a pet project of mine and one that I'm keen to continue ticking over. So Ubuntu Mate on the Pi is a little different [to the PC version]. It's the same base but it's got all of the Python stack and all of the GPIO and all of the SPI and I2C and Scratch and MineCraft Pi Edition – all of the things that people expect to be on the Pi are on the Ubuntu Mate Pi version.

The PowerPC port was the same. I didn't create that. Someone in the community did the legwork and then I was simply able to adopt their work into the official build system. All the architecture ports have come from the community and I've adopted them and made them official.

Were the and major technical challenges in getting Ubuntu Mate to run on the Raspberry Pi?

MW: Yes – Bluetooth was a complete pig to get working. I had access to a guy called Phil Elwell at the Raspberry Pi Foundation over the course of those four days leading up to the launch and he was providing me with kernel patches, firmware blobs and what-have-you. The Wi-Fi enablement came up within minutes – I was really pleased with that – but getting Bluetooth to work was a protracted experience. There are some complications… Bluetooth in the Pi 3 is a H5 device [a low-level communications protocol] but at that time they couldn't get the H5 mode to work and they butchered it to work in H4 mode with no flow control at all – that then requires that you patch Bluez quite heavily. It's an incompatible patch set so by building it for the Pi 3 you effectively break Bluetooth for every other device out there as it's very hardware specific. I now know more about Bluetooth than I'll ever need to!

What can we expect in the future of Ubuntu Mate?

MW: I've been interested in snaps [Canonical's container-based packaging format] since I first heard about them. I've been poking people at Canonical saying "is now the time to do more with this?" and for a while it was, "No, this is very fast moving, very experimental, just learn about the concepts, don't actually do anything with it", but around the end of last year, the very beginnings of this year, I started poking Will Cook, the desktop manager at Canonical, and saying "I'm really interested in doing more with snaps, what is it you're planning?" I got involved in the Snappy play pens quite early on – making

"Comparing KDE Plasma, Gnome 3 and Unity [on the Raspberry Pi], Unity works the best." Shocking!

snaps for desktop applications – and then I've been on the Snappy sprint. Now I've got a good understanding of what that road ahead looks like.

Probably not for 16.10, because there's a lot of work to do for that anyway, but for 17.04 onwards, I would like to see snaps playing a significant role in Ubuntu Mate's future. Our tag line is "for a retrospective future": the retrospective bit is about the classic desktop environment, but the future is that there are new technologies and we have to move with these. We can keep our traditional paradigm but we have to move with the new technological movement – we can't just be languishing.

What we have now is a classic Ubuntu system that can install and run snaps, but alongside that, I'd like to work towards having a fully snapped desktop environment. If we can offer both, you can choose a classic install or a snapped install. Where the snap installs really come into their own is on things like the Raspberry Pi where you can chain what are called gadget snaps and deliver your kernel, your base operating system and your application stack all as snaps and have atomic updates across the whole thing, particularly on ARM devices where you're targeting a very precise hardware architecture and configuration options.

In this setup, would the Mate desktop fit in its own snap?

MW: Yeah, so this was something we started on at the sprint. A new concept that's been introduced into snap packaging is platforms – in the vernacular of Flatpak they talk about runtimes. Robert Ancell (who's part of the desktop team at Canonical) has started on a *GTK 3.20* platform snap and there's a new content interface in snaps that enable you to do content sharing, which basically says "in order for this application to run, it requires this platform snap." Instead of having to duplicate all the common dependencies within every snap (this is one of the criticisms that snaps are larger than they need be because they all carry the same stuff over and over again), you can now say that they need this platform snap and you don't have to carry all that common baggage in each of your snaps.

> I'd like to work towards having a fully snapped desktop environment

"As a spin off of making Ubuntu Mate for the Raspberry Pi, I made a build system that can build all of the Ubuntu flavours for the Raspberry Pi. There's a website called ubuntu-pi-flavour-maker.org where you can download images for the Ubuntu flavours."

What are the big advantages that you get from snaps?

MW: The main one for me is that as Mate upstream, we're already making tarballs with build systems in them, so we're delivering packages, so of course every distribution under the sun then has to repackage that. We all sit there, we make these tarballs and test these tarballs and within the team we've got people from Fedora, Gentoo, Slackware, Arch, Debian, Ubuntu, Linux Mint and some of the more esoteric things like Vector. We all go off and package this stuff again for all of these different distributions and it would be good if we as the upstream we could say 'here is Mate desktop' – whatever version – and we're able to just deliver it lock, stock and barrel and everybody gets it. We can deliver the upstream vision and everyone can have it and we can spend more time working on the future rather than us all reimplementing the same packaging all over again. That's one key thing I really want to see solved.

The other thing is the update and rollback, and being able to have multiple different versions of the same runtime. Sometimes there's a version of an application that's specific to a version of Mate and as we lockstep move the desktop along, that application that hasn't been updated doesn't work any more and it would be nice to still have those things work – snaps enable us to to that.

It sounds like there are some quite big releases coming up for you with *GTK 3* and then snaps.

MW: Yeah. we've done the big work with *GTK 3* and the snaps will keep us busy for a little while. Western Digital are talking to us about making an Ubuntu Mate device a bit like they have with OwnCloud (and they're also talking to NextCloud), but that presents some difficulties with *VLC* (which we ship by default) as they need to know what their exposure is with the codecs and licensing that they need to cover from that point of view. *VLC* is snapped so we could stodge that and put the *VLC* snap in the software boutique and then they don't have to distribute anything that does incur any licensing difficulties for them.

On single-board computers in general, Snaps are are going to be very strong indeed, particularly when you look at what OpenWRT has done supporting both paradigms– that's where snaps are really going to win.

BUY LINUXVOICE MUGS AND T-SHIRTS!

shop.linuxvoice.com

REVIEWS

The latest software and hardware, rigorously bashed against a wall by our crack team.

Andrew Gregory
Has added a new machine to the PC graveyard: his main Linux box. Time for a new one.

Having just got into the magical world of JavaScript programming, I've come across the Atom text editor. I know I'm late to the party (especially so given that we reviewed *Atom* on these very pages in issue 18). Its killer feature is that it's infinitely customisable via CSS.

And, true to form, I haven't customised it one bit. The flexibility built-in means that it has already passed in front of many more aesthetically attuned eyes than mine, so it's better that I could ever make it. In contrast, interfaces that can't be modified (Windows 10) or that theoretically can be modified but practically are just too damn confusing (KDE) will always look bad. I don't need to do anything to make Atom better; it matters that I am empowered to do so.

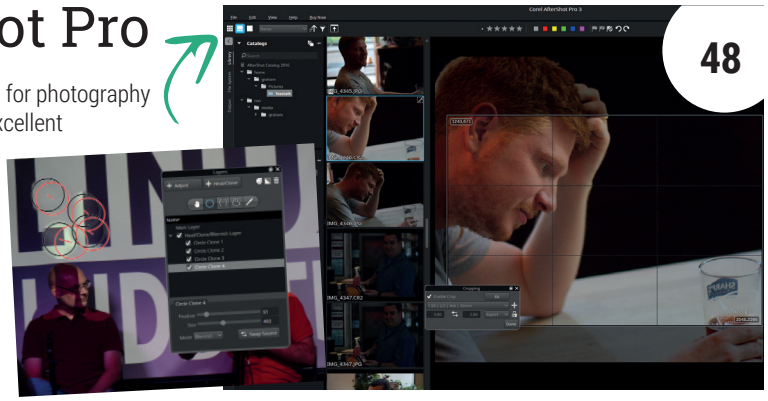
Extend this logic to web browsers, image editors, emissions cheat devices in Volkswagen cars *et al* and you reach the point of why Free Software is important. I'm never going to learn to write a desktop interface so that I can make it more intuitive – but I appreciate that, in theory at least, I could do.

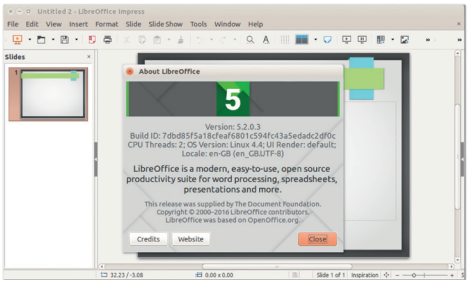
andrew@linuxvoice.com

On test this issue . . .

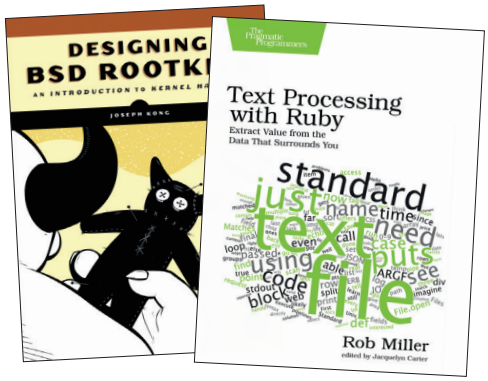
Aftershot Pro

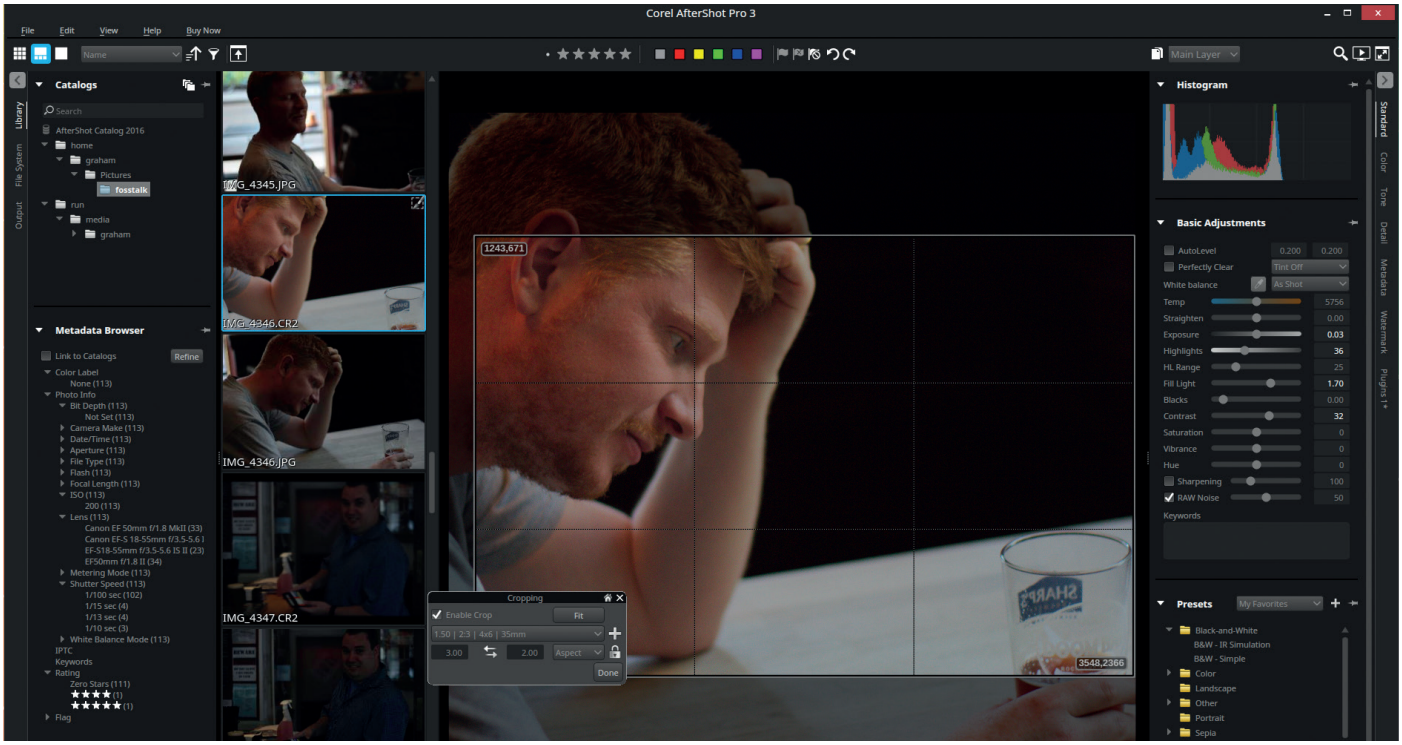
What a time to ba alive for photography geeks on Linux. This excellent RAW image editor is non-free (boo!) but is chock-full of incredibly useful filters and other effects. Hurrah!





Group test and books





AfterShotPro 3

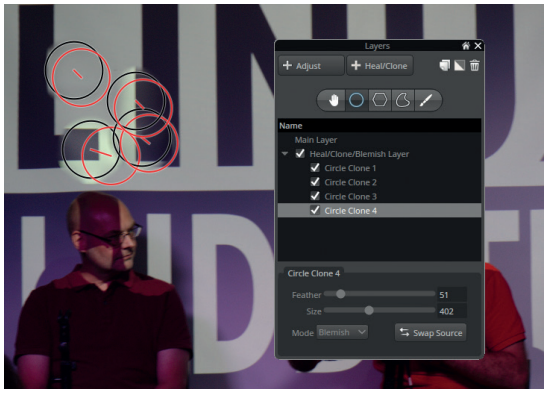
Graham Morrison avoids years of photography theory by using software.

Web www.aftershotpro.com
Price £69.99

AfterShot Pro is a product with its sights on Adobe's equivalent *Lightroom*, which is now close to being an industry standard at enabling photographers to manipulate the RAW image data output from their digital cameras. Both these applications let you make poorly taken photographs, such as those taken in bad light, with the wrong white balance, and with huge areas of contrast, look awesome, and good photographs look spectacular. What *AfterShot Pro* does better than the open source tools we've tried is use intelligent algorithms to make these adjustments easy.

The combination of 'Back Fill' with *AfterShot Pro*'s new highlight recovery tools is the perfect example because they can solve all kinds of common problems with just three sliders, from under-exposure to lack of detail. Unlike adjusting brightness in *Gimp*, however, these sliders are also intelligently adjusting shadow and mid-tone exposure, cleverly balancing the processing within the image to keep everything looking as natural as possible. We've found it works much better than the equivalent tools in the open source options, and is a genuine rival to similar facilities in *Lightroom*. The new blemish removal tool also works well, and stops you switching applications, which is almost impossible if you want to keep working within the RAW domain until the final export.

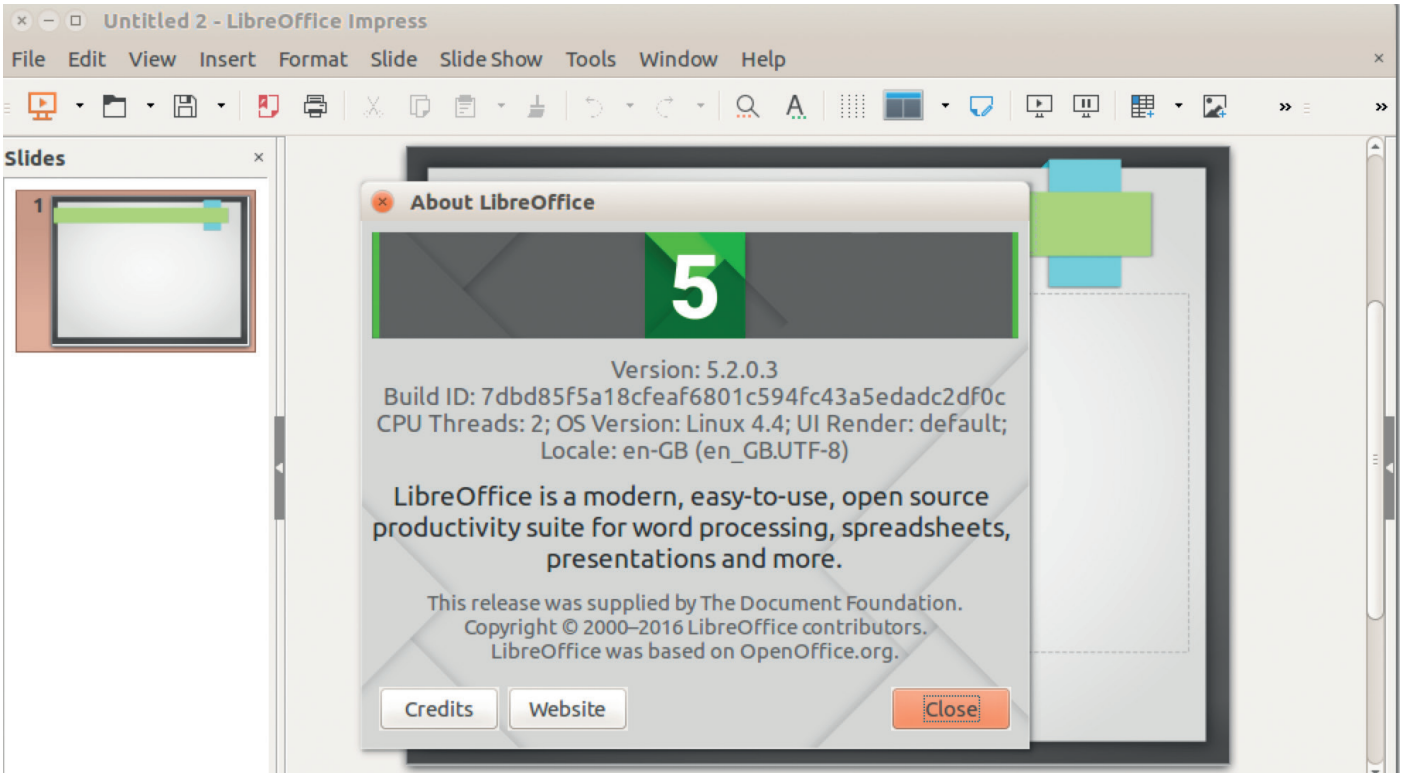
This version also promised to be a lot quicker. We needed to install the OpenCL drivers for our graphics hardware, but after this, image rendering was snappy – much quicker than the previous version, and we can imagine better hardware with better OpenCL acceleration would be very quick indeed. On the down-side, we didn't find any options to force the GUI to adapt to our high-DPI display, which is a promised feature for other operating systems. If you're a professional photographer, it's likely you need support and a solid application for managing your library of photos and processing them for clients. You may even be tempted to move to Linux after Adobe recently denied standalone *Lightroom* users access to updated features because of 'SOX and accounting practices'. For these reasons *Aftershot Pro* is worth paying for.



The new Blemish tool means you don't need to do any final touch-ups with *Gimp*.

A powerful, if non-Free, professional photo manipulator.

★★★★☆



LibreOffice 5.2

Graham Morrison investigates the new release of the premier FOSS office suite.

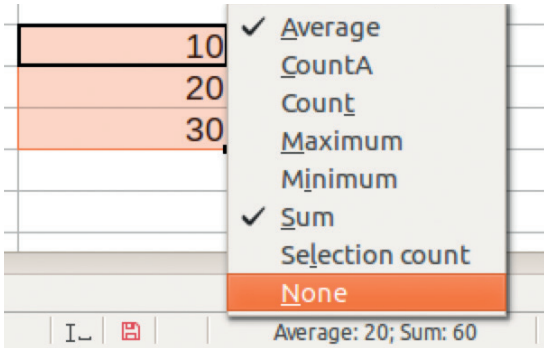
Like clockwork, *LibreOffice* 5.2 arrived in early August. Whereas the 4.x series of the suite focused on cleaning up the code and other "under the hood" improvements, in 5.x the developers have paid more attention to the interface, tweaking menus and toolbars for a smoother user experience.

So what's to shout about in *LibreOffice 5.2*? For enterprises, there's a new document classification system based on standards from TSCP, the Transglobal Secure Collaboration Program. This lets you mark a document as internal or classified, with an appropriate warning and watermark. It's worth noting that development on this feature was supported by the Dutch Ministry of Defence – *LibreOffice* is really making inroads in government departments.

More than just .txt

There's also support for multiple document signature descriptions, along with import and export of signatures from OOXML files (as used in *Microsoft Office*). In terms of user-facing changes, *Writer* and *Calc* now have single toolbar modes, while *Calc* has new forecasting functions and the option to enable multiple statistic views in the status bar. Some dialogs have been redesigned to be easier to use, and extra buttons have been added to toolbars (eg a currency drop-down selector in *Calc*) as well.

Behind the scenes, performance in OpenGL and OpenCL has been improved, unit tests have been added, and a new crash reporter tool (for Windows) should help to narrow down bugs. Then there are the usual updates to file-format compatibility – docs from other suites should render better in this release.



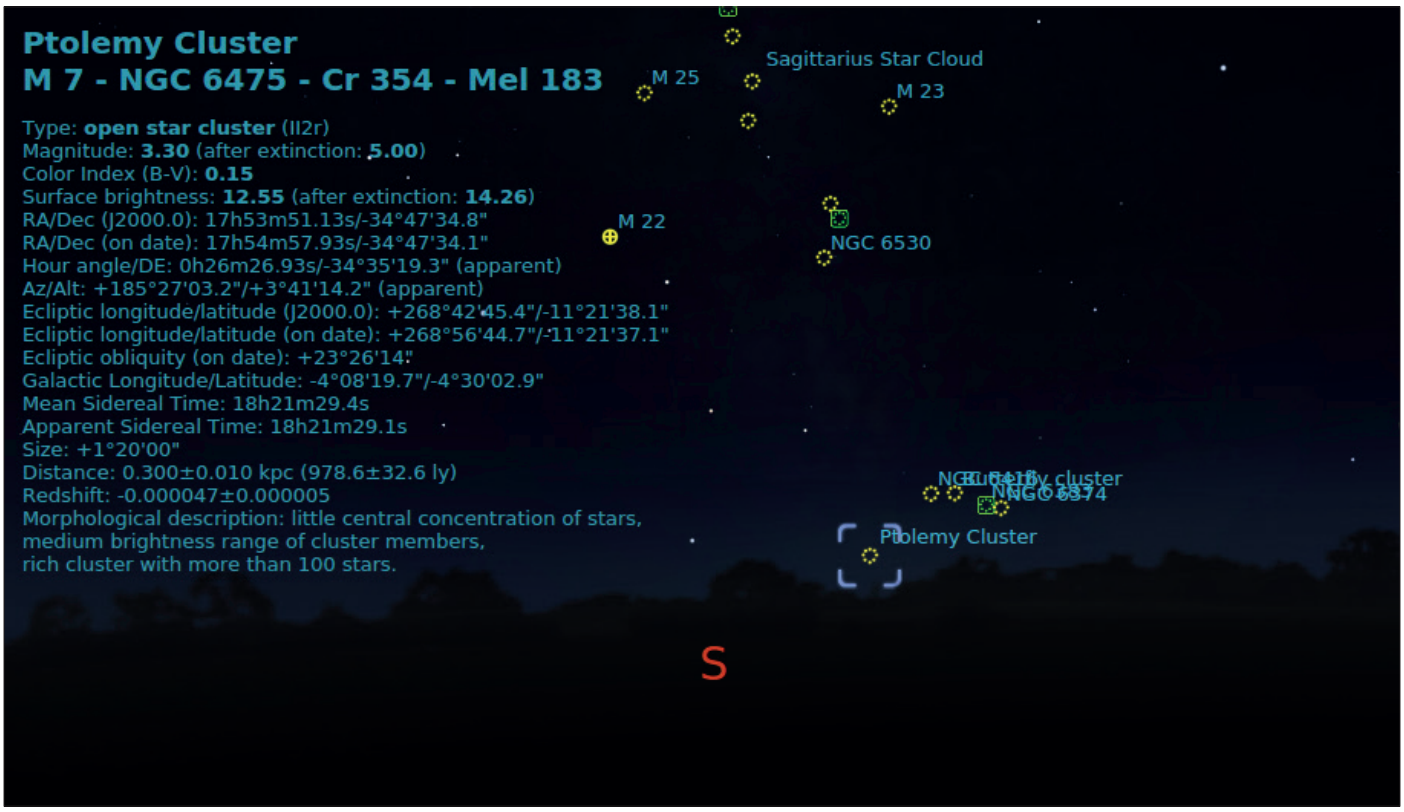
In *Calc*, the status bar can now show multiple functions – useful for getting a quick overview of data.

LibreOffice remains the flagship FOSS office suite, but it really could do with an online "cloud" version. Progress is being made on this front thanks to Collabora, and it's possible to use a limited version of *LibreOffice* inside *Own/NextCloud*, but it's early days. Hopefully for the next release of *LibreOffice* in February 2017, it'll be quick and simple to install on a server and access anywhere in a browser.

Web www.libreoffice.org
Price Free
Licence MPL v2.0

Not a world-changing release, but taking a step in the right direction with lots of small refinements.

★★★★☆



Stellarium 0.15

Ben Everard lies in the gutter, gazes up at the sky and wonders what he sees.

Website www.stellarium.org
Developer Fabien Chéreau and contributors
Licence GPL

Gaze upon the night sky and you'll see thousands of specks of light. If you're lucky enough to have dark skies, and bored enough to spend the night counting, you should be able to see around 5,000 stars at a time. These stars, with a few minor alterations, have been the backdrop of history. They're the same stars that guided Christopher Columbus across the Atlantic, Ernest Shackleton across the Antarctic and Vasco da Gama to India.

The options windows appear in a dark theme so they don't ruin your night vision.

Even in the space age, the importance of the stars cannot be forgotten, as they shine on long after the batteries in an electronic navigation system run flat. When lightning struck the Apollo 12 spacecraft as it launched from Kennedy Space Centre and knocked out the power to the command module, it was the position of the stars that guided the crew safely into orbit and on to the moon.

Feel like some watcher of the skies

The ancients saw in the stars the figures of their gods, while even now, in more prosaic times when we know that they're giant balls of hydrogen gas slowly fusing into helium, we can't help but wonder if one of those pin-pricks of light has orbiting around it a giant rock with some creature upon it gazing back at us.

Stellarium is an interactive guide to the speckled blue-black dome that hangs over our nights. Start the application and you'll get a full screen view of the sky in your current position (or as close as *Stellarium* can tell) at the current time. Move your mouse to the bottom half of the left-hand edge of the screen and you can change the location and time to see what the sky will look like at any place at any point in time (either in history or in the future). This makes *Stellarium* a great tool for planning your stargazing – just enter the place and time and you can see what you'll be in for in real life. The real advantage is not that

you get a sneak preview, but that you can learn a bit more about what you'll see. Move your mouse to the left-hand side of the bottom of the screen and a menu will pop up that enables you to select what you want to display. The first three buttons from the bottom-left-hand corner are Constellation Lines, Constellation Names and Constellation Art. With these switched on, you can get a better understanding of the skies, at least from an ancient Greek perspective.

It's not all Greek to me

There's a huge number of things in the night sky, and only a fraction of them are visible with the naked eye. If you have a pair of binoculars or a telescope, you can explore far more than just the constellations. To change what appears in the sky, press F4 to open the Viewing Options window. In the DSO (Deep Space Objects) tab, you can configure the various items that you want to appear. Check the Labels And Markers box and *Stellarium* will highlight the most visible of the displayed objects in the sky, and you can click on these in the main display to get more information (including the details to position a telescope at that item). In the Sky tab of the Viewing Options window, you can change various settings to make the interface a little more captivating. If you increase the twinkle setting and the number of shooting stars, you can create a backdrop that's more magical.

Version 0.15 of *Stellarium* brings support for remote controls, which greatly enhances the ability to use the software for planetariums and other displays. It brings in more data in the form of support for more DSO catalogues and more folklores. There's more to constellations than the Greek one, and *Stellarium* now supports 26 cultures' views of the stars. Using this, we discovered that the Inuit describe the constellation we know as the Great Bear (Ursa Major) as the Caribou. It is, though, apparently just this reviewer who knows it as the frying pan. Orion, meanwhile, is known as the

If you zoom right out you can see the entire sky in a single view.

Plougher and Oxen to the people of Macedonia. There is no other subject that knits together science and folklore quite as well as the night sky, and *Stellarium* does a great job of catering to both areas of interest.

There is some software that you don't realise you even wanted until you try it, and *Stellarium* falls into this category. Even if you've never thought you wanted to zoom in on the finer points of the stars, we recommend you grab this and get better acquainted with the night sky. Now, please excuse me, it's getting dark outside and Aquarius should be rising over the back hedge soon.

> Stellarium is an interactive guide to the speckled blue-black dome that hangs over our nights

Discover the universe with the best interactive guide to the night sky.

★★★★★

The artwork helps you understand how the constellations got their names.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

TRADE FOLLOWS THE FLAG

Michel Loubet-Jambert is our Games Editor. He hasn't had a decent night's sleep since Steam came out on Linux.

Both AMD and Nvidia have released a new generation of graphics cards, which have huge performance improvements over the last. Linux was not left out in the cold by either vendor as the updated drivers which support these cards were pushed out straight away.

On the other hand, users of this hardware will not see as many performance gains as Windows users given that most games are not as well optimised for OpenGL as they are for DirectX. There are signs of this getting better as more developers create games with cross-platform support in mind while using engines that are also constantly improving their OpenGL and Vulkan support. It's mostly the older games created with a Windows-only development mindset that have the biggest performance gaps. On the Intel side of things, Vulkan support from the Ivy Bridge architecture onwards has been pushed out as part of the Mesa 12 release, which also saw OpenGL 4.3 support and other big changes.

Meanwhile, the itch.io game store has brought out a new version of its open source store application, with improved Arch Linux support and much better stability in what has become a very mature application. This is a great alternative to Steam for those looking at less commercial games, especially considering the delays in GOG's Galaxy application arriving on Linux. Itch.io has been making some changes recently, including the "refinary", which is similar to Early Access.

Undertale

Does the indie smash hit live up to the hype?

Web http://store.steampowered.com/app/391540
Price £6.99

It takes about 30 seconds to realise that *Undertale* is very weird. That's probably what made it such a smash hit with users of YouTube's Let's Play services, which hepled it sell close to 2 million copies, a figure which many games with multi-million dollar budgets would struggle to hit.

The game is an 8-bit RPG with a twist, in that the player can choose to fight or make friends with enemies. Making friends means that monsters of the same type can be avoided in the future, but also voids any experience gain, a serious drawback in a situation where there is no choice but to fight. Friendship can be established through talking to, petting or even flirting with enemies, which is all part of the game's humour, often toying with RPG conventions.

At first glance, *Undertale* is a very ugly game more reminiscent of *MS Paint* than charming 8-bit nostalgia. In most games, this would put a lot of people off straight away, but the fact that it manages to keep players interested is testament to a game where nothing is superfluous. The characters encountered in the dungeon are humorous, each with their own personality and traits shown through dialogue and during the game's odd minigame battle system.

Looks aren't everything, and *Undertale* surprises in many ways beyond being better than it looks.

The game isn't all perfect though: its environments can be bland and its puzzles are often tedious, but this can be overlooked since it keeps on surprising throughout. *Undertale* is a unique experience that we highly recommend.

***Undertale* can convey more in a few pixels than many AAA games can in their entirety.**

Haven Moon

A spiritual successor to Myst.

Web http://store.steampowered.com/app/493720
Price £10.99

It's been over 10 years since the last installment of the hugely popular *Myst* series was released, so it seems that now is good time to satisfy people with that particular itch. Even on a purely visual level *Haven Moon* gets the nostalgia receptors tingling, though in fantastic real-time 3D. It's a one-man development effort, and the love for the *Myst* games really comes through.

The sense of intrigue is a prominent theme, as the player is dropped into a solitary and mysterious world full of machinery and hidden passages. This experience is further enhanced by the story, which begins with the player finding a note from the survivor of a cataclysmic event, wishing to leave behind his life's treasure. *Myst* fans are sure to enjoy this game, but it's also a good game for those wanting to delve into the series, but put off by the dated visuals or wrestling with *DOSBox* configurations.

***Haven Moon*'s graphics manage to take the player back to 1993 while still looking modern.**

Day of the Tentacle Remastered

A point-and-click classic brought back to life.

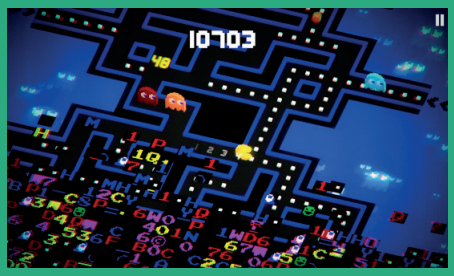
Web http://store.steampowered.com/app/388210
Price £10.99

The early to mid 1990s is often seen as a golden era for point-and-click adventure games, with *Day of the Tentacle* being one of its prime examples. Though this game and others have been playable up to now thanks to the open source *SCUMMVM* software, this version features completely redrawn high-resolution artwork (no more upscaled 320x200) and remastered audio, bearing in mind the original non-CD versions of the game had no voiceovers. A nice touch is the ability to switch between the old and new visuals and audio, to get a more nostalgic experience or simply compare old and new to see how much work has been done to the game, similar to the *Monkey Island* remasters some years back. Unusually though, a hint system has not been added, despite it being the norm in contemporary adventure games.

The new visuals are great for those who like retro games without dated graphics.

For those not aware of the story and premise, *Day of the Tentacle* is the sequel to *Maniac Mansion*. There's a lot of time travel involved and constant gags through the game's three varied protagonists and excellent supporting characters, making this a must-have for any fan of the genre.

ALSO RELEASED...



PAC-MAN 256
***Pac-Man* is officially on Linux! The game has held up extremely well over the years so younger games shouldn't dismiss it, more so given the fact that the PC version lacks the pay-to-win nonsense of its mobile counterpart. The 2.5D graphics, power ups and unlockables give enough new reasons to play while not sacrificing what made the game great.**
http://store.steampowered.com/app/455400

Kelvin and the Infamous Machine
After adding some voice acting during a brief Early Access period, *Kelvin and the Infamous Machine* has been released, offering a solid point-and-click game revolving around time travel. In the game, Kelvin zips around history, encountering the likes of Leonardo da Vinci and Isaac Newton while solving their dilemmas. The game is short, but has nice visuals and humour typical of the genre.
http://store.steampowered.com/app/376520

The Ship: Remastered
This very eccentric multiplayer game is currently in Early Access, and is mostly bug-free. The game throws most of the conventions of the multiplayer shooter out of the window and instead puts players on an old timey cruise ship in a "whodunnit" situation. The player must find out who they must murder and track them down, all the while escaping from their pursuer. There's plenty of laughs and very original gameplay.
http://store.steampowered.com/app/383790

Designing BSD Rootkits

Ben Everard uses rootkits to avoid having to remember SSH passwords.

Author Joseph Kong
Publisher No Starch Press
Price £21.50
ISBN 978-1593271428

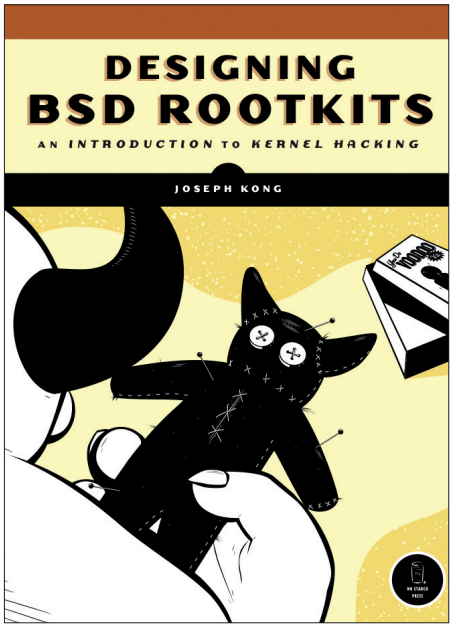
Rootkits are bits of software that attackers use to maintain access to a computer after they've broken in. Essentially, they're software kits to maintain root privileges. A good rootkit has to be invisible to the system administrator yet at the same time have complete control over the machine in question, and the best way to achieve that elusive goal is by running in the kernel.

Designing BSD Rootkits is a book about how to control a running BSD kernel to allow you to nefariously manipulate data – including core kernel data such as the list of open ports. This is obviously a massively complicated subject, and in order to get the best out of this book the reader is expected to be a reasonably competent C programmer with some knowledge of assembly language.

The main aim of this book is obviously to help penetration testers and security researchers understand how to build rootkits for BSD, a Unix-alike alternative to Linux. There is also a secondary use – in order to build a rootkit, you have to have a solid understanding of how the kernel works, and a great way to get this understanding is to poke it in ways it's not designed to be poked. In other words, it makes a good, if highly unorthodox, introduction to the BSD kernel (and, indeed, all Posix kernels).

An unusual and interesting way to start investigating Unix kernels.

★★★★☆



No Voodoo Beasties were harmed in the making of this book.

Text Processing With Ruby

Ben Everard starts work on a magazine editing robot in the hope of retiring. One day!

Author Rob Miller
Publisher Pragmatic Bookshelf
Price £25.50
ISBN 978-1680500707

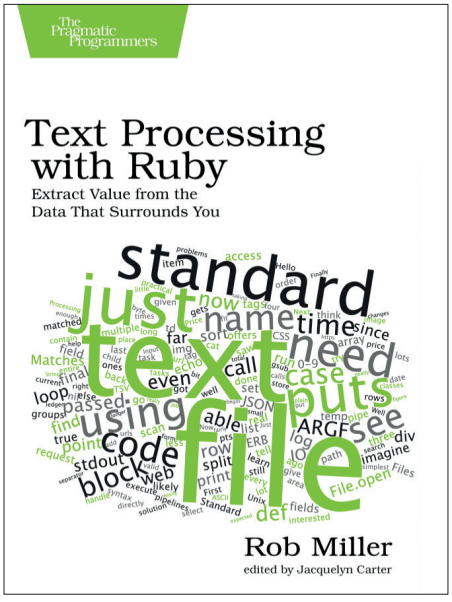
Text processing is one of the most useful skills to have in programming. Whether it's pulling data out of log files, scraping websites or analysing the sentiment in millions of Twitter posts, the ability to condense a large amount of ASCII (or Unicode) into a useful form is vital. Text's importance comes from the fact that it is the bridge between computers and humans – it's the only format that both can easily understand. It's sometimes called the universal format and shows no sign of becoming less popular despite being one of the earliest forms of data representation.

Text Processing With Ruby does a good job of covering the whole area of modern text processing by splitting the subject into three parts. The first part of the book looks at acquiring text including reading from files, getting standard input and scraping websites. The second looks at transforming text, which comes down to regular expressions, parsers and natural language processing. The final chapters explore writing text to files and standard output.

This book is thorough, informative but a little unexciting. There are plenty of examples to help the reader understand what's going on, but they're all a little contrived. More meaty examples would help keep us more engaged and help us see exactly how everything came together.

A thoroughly useful book for new Ruby programmers, but it lacks pizzazz.

★★★★☆

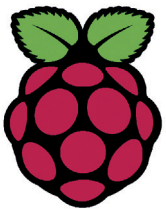


If you only learn one programming skill, learn text processing.

LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM



FREE SOFTWARE | FREE SPEECH

GROUP TEST

A serial online socialiser, **Mayank Sharma** would do anything to avoid real work – even test a bunch of web apps to deploy his own social network.

On test

BuddyPress

URL https://buddypress.org
Licence GNU GPL v2
Latest release 2.6.1.1
Is this the WordPress *for social networks?*

Community Builder

URL www.joomlapolis.com/community-builder
Licence GNU GPL v2
Latest release 2.0.14+
We know what it does, but is it any good?

Elgg

URL www.elgg.org
Licence GNU GPL v2
Latest release 2.1.2
Will I let my familiarity with the software cloud my judgement?

Mahara

URL www.mahara.org
Licence GNU GPL v3
Latest release 16.04
Can it bid Sayonara to the rest?

Oxwall

URL http://developers.oxwall.com
Licence Common Public Attribution Licence
Latest release 1.8.3
Will it make the rest drive up the wall?

Trident

URL www.online.me
Licence Creative Commons Public Licence
Latest release 8.0
Can it poke the competition?

Create your own social network

An online social network operates pretty much like its physical offline counterpart: you meet people, establish connections, keep up with the contacts, and continue the relationship. But social networking on the internet has one major benefit over offline ones – it enables you to find like-minded people beyond your physical network, even across time zones. Also thanks to the power of keywords and search fields, you can skim through the noise and find like-minded groups from the comfort of your armchair.

But why should you go through the pains of hosting and managing your own social network when you can join one instantly for free? For starters, hosting your own social network gives you the option to customise and brand it as per your whims and wishes. This is also helpful if you want to integrate the social network into your existing online infrastructure.

The web apps in this group test help you create focused social networks as a means of connecting people. You can deploy them to bring together students and equip them with the means of collaborating on projects, exchanging class notes and even advertising the availability of dorm rooms. For businesses, a custom social network can be an ideal extension of a bulletin board or a company intranet. It also helps bring together physically separated people who are connected through a strong common thread, like the various campuses of a university, or regional offices of a multinational corporation. Similarly, a corporation could deploy such a social network on its intranet as a virtual water cooler for its employees and a means for them to exchange notes.

> Hosting your own social network gives you the option to customise and brand it as per your wishes

Deploy a server

Unlike standalone desktop apps, the web apps tested over the next few pages are designed to reside on a web server that you'll have to set up before you can fiddle around with any of these apps. LAMP is the most common platform for rolling out server software. It uses a Linux distribution as base that has the *Apache* web server together with the *MySQL* database, along with support for displaying apps written in PHP or Perl. While it doesn't take much effort to roll out a web server, the exact procedure depends on the Linux distribution that'll power it.

Most web apps will work atop the LAMP platform; some might even allow you to swap out components for other alternatives. All apps have detailed installation instructions. The installation document will also list the exact libraries that they require and should be one of the first documents you read if you want to try out one of these web apps.

What makes a social network?

Tools of the trade.

A social networking website is a Web 2.0 service. The term Web 2.0, coined by O'Reilly Media in 2004, isn't a technology update to the internet, but rather refers to the so-called second generation of web-based services that encourages reader participation. A social network is made up of several such communication tools and services, including blogs for sharing content and forum boards for fostering discussions.

Furthermore, there are certain elements, without which social networking on the web wouldn't be possible. User profiles play an important part in helping you find like-minded people. Once you've found the people you want to be connected to, the network should enable you to connect with them and add them as friends. After connecting with friends, you'd want to share information, ideas, pictures, or documents with them, which is another important aspect of an online social network. To facilitate the discovery of shared information, the network should have the provision to assign keywords or tags to them. In fact, tags are so popular that they've paved the way for Folksonomy or collaborative tagging. By transferring indexing control to users, Folksonomy websites incite people to share. To keep its members updated, the network should also have some form of feed syndication.

Oxwall

Your own Facebook.

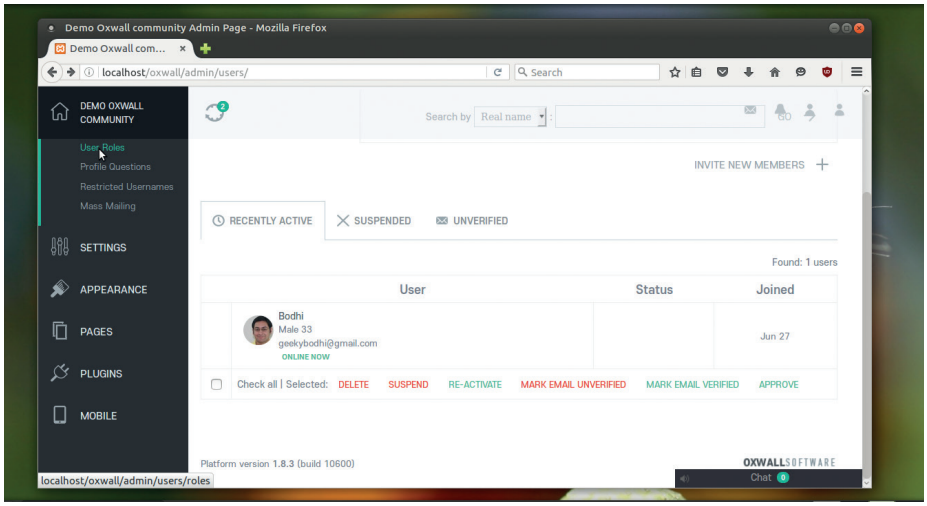
Oxwall ships with all the essential components you need to get users to interact. Users can blog, share pictures and videos, discuss via forums, share with wikis, comment, tag, and rate all types of content, and a lot more. The core software and the essential plugins are available gratis, while additional plugins can be bought for a one-time fee. *Oxwall* relies on a simple step-by-step installation process. At the last step the installer displays a list of plugins that you can enable. These can be activated or disabled later from the administration console.

Oxwall's admin interface is pretty well laid out and divided into various sections. While you edit your network, ensure it's not visible to anyone by disabling the site and putting up a maintenance page with a personalised message. If you are running a closed network, or a very specialised one, you might want to make the casual visitor pause and read about the network they are about to enter by putting up a splash screen.

Make it your own

The most convenient way to personalise your network is to alter the default theme. *Oxwall* ships with a bunch of themes, and you can grab some more from its website (**http://www.oxwall.org/store/list/theme/latest**).

You can tweak various aspects of the theme right from within *Oxwall*'s admin interface. Depending on the theme, it might also take several images, such as for a background and logos, and you can replace them by uploading custom graphics. The theme customiser also includes a CSS tab, which offers fine control over the CSS style of the theme.



You can easily monitor active users and maintain decorum by suspending abusive users.

Broaden the horizon

In addition to customising the network visually you can also make dramatic changes by activating plugins.
Some plugins, such as the Activity Notifications and Friends plugins, are pretty straightforward and require no configuration. But plugins such as the Forum plugin and the Contact Importer offer some additional controls. For example, to import contacts from Google or Facebook, you need to equip *Oxwall* with your Google Client ID, and Facebook App ID.

Besides the additional plugins that ship with the app, you can fetch additional ones from the *Oxwall* store (**http://www.oxwall.org/store/list/plugin/featured**). Some plugins are only available for a fee, such as the PayPal Billing plugin, which lets you accept payments from users of your network via PayPal.

Oxwall also lets you rearrange the default layout of the dashboard depending on the plugins that you have selected. You can rearrange the pages and menus by dragging and dropping with the mouse. You can also edit each item to either make them visible to everyone, to guests, or to every logged-in member. You can enable users to customise their own page components, but you can "freeze" certain elements to prevent them from being edited or moved by the users.

Control your users

Managing users is one of the strong points of *Oxwall*. Before you throw open your network, you should create custom account types and edit profile questions for each. For example, if you're setting up a network for an educational institute you can have different registration questions depending on whether the user is a teacher or a student. If this all sounds a little daunting, head to **demo.oxwall.com** to get to grips with the administration of the network.

Helps roll out a feature rich network that can be customised to the hilt.

★★★★☆

Mahara

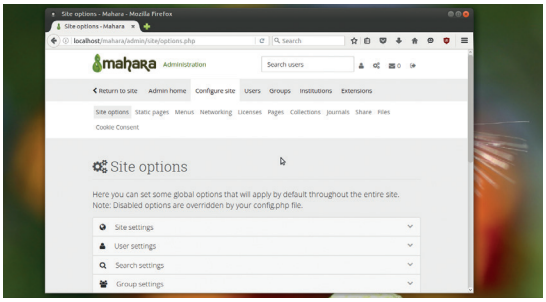
The LinkedIN for education.

Unlike the other social networking software in this group test, *Mahara* caters to a particular demographic. It's designed to allow users to demonstrate their skills and work by creating digital portfolios. Additionally you can also use the platform as a social networking system to enable users to interact with their friends and create their own online communities. *Mahara* includes tools and services such as blogs, custom pages, a resume builder, and can also be hooked with the Moodle learning platform. *Mahara* is a popular tool among universities all over the world that use it to record achievements and assist with personal career development of their pupils.

While installing, *Mahara* shouldn't throw any unexpected issues, don't make the mistake of using the software without first spending some time with its documentation to grasp its true potential: Mahara ePortfolios, which can hold everything that can be stored digitally. These components are individually known as artefacts. Multiple artifacts are collated into pages and you can have as many pages as you like, either with a different number of artefacts, or different purposes and audience. For example, you can have a page for your family that includes holiday photos. Similarly, you can create a page to showcase your work and a resume to potential employers. If you want people to see your page you can add them as individuals or as a member of a group, and pages can even be made publicly available.

Admin panel

Besides the Dashboard, *Mahara*'s administration interface includes tabs for uploading and editing content, arranging them into portfolios, and defining groups. Creating a portfolio is a fairly intuitive process once you get the hang of the administration interface. You can choose the layout of the page and drag and drop everything from blocks of plaintext to recent forum posts, PDF files, images and a lot more. *Mahara* hosts a demo (**http://demo.mahara.org**) to tinker with its back end.



You can export your portfolios as HTML sites or in a format that can be imported into another Mahara install.

Helpful for creating a network that shows off the user's achievements.

★★★☆☆

Trident

A good beginning.

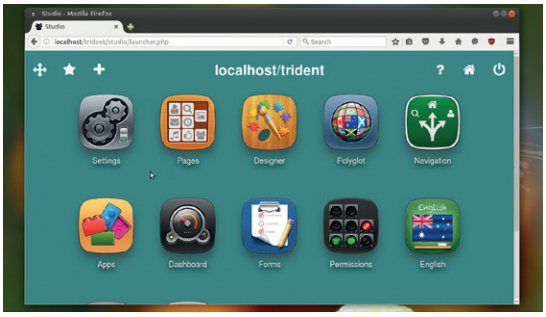
Trident bills itself as a social CMS, and had its first stable release late last year after half a decade of development. It offers several unique features, such as the ability to create multiple profiles for a user's different personas. *Trident* has the usual slew of social networking features, such as the ability to make friends, communicate with them, share and like content, and more. To the admin it offers tools such as custom page builders, navigation menu builders, permissions control and several other common content management functions. The developers boast of testing the app on all form factors and claim that its typography, link targets and navigation are all optimised for touch devices.

Before you install *Trident*, you'll need to register for a free account with its commercial developer BoonEx and grab a set of keys. You'll need these to download additional apps and extensions (both free and paid-for) from *Trident*'s marketplace. The software is made up of two main components. There's a content management interface for managing the installation, which can be handled by appointed admins and moderators. Then there's the Studio, which helps you design and extend *Trident*'s functionality.

Initial minimalism

There aren't many components inside Studio on a fresh installation. You get a tour of the default core components and can start building your network by customising them, such as your profile, the site's homepage and other pages. You can then download more apps from the marketplace, which lists free and paid-for apps.

Studio makes it straightforward to customise your *Trident* installation. For example, you can build a page by picking the right blocks and configuring their visibility for different members. Then there's the designer, which helps you change the site's logo, design and manage the cover image as well. Once you get familiar with *Trident*, you can use the forms tool to customise *Trident*'s database and forms.



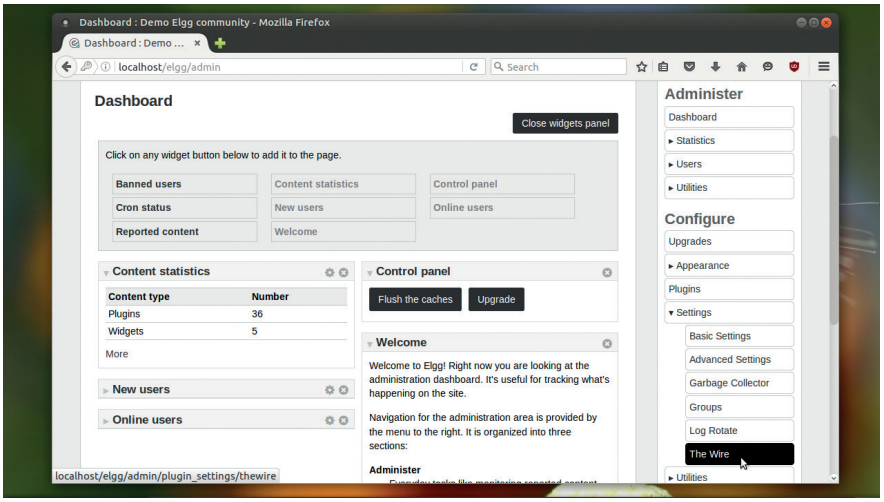
You can use the pages module to customise the content, visibility settings and layout of a page.

One of the easiest networks to configure, but it ships with a limited set of functions.

★★★☆☆

Elgg

Old but still going strong.



Elgg users can also use the built-in messaging capabilities to send messages to other registered users on the network.

Elgg is one of the most versatile pieces of software that offers almost everything you need to start a social networking site such as a blog, filesharing features and a Twitter-like service. Setting it up is a pretty standard affair.

The administrator dashboard divides the settings under two major heads, namely Administer and Configure. The options under the Administer category give you various statistics about the server, lets you view online users, add new users and put the network in maintenance mode while you're setting it up. The Configure category houses quite a lot of functionality. The Settings section in particular is useful for altering options that affect the entire *Elgg* deployment. From here you can change the name of your social network that you specified in the wizard while deploying *Elgg*.

Customisation galore

The default *Elgg*-fuelled network is very barebones. You can start by customising the order of the menu items displayed at the top and can also add custom menu items. You can also modify the default profile fields. If the existing items in the profile don't work, you can easily replace them with something that suits your requirements, and can even create your own custom fields.

For more flexibility in designing a custom profile page you can use the popular Profile Manager plugin. If you wish to revamp the front page, enable the Front Page Demo plugin. *Elgg* ships with over 30 plugins, such as blog, bookmarks, pages, notifications, etc. Some of the plugins that you might want to activate includes Site Pages, which lets you create simple web pages, and Tag Cloud for displaying all tags. Furthermore, *Elgg* has an active plugin community that churns out plugins by the hundreds. In the *Elgg* world, themes are also treated as plugins. Some of the popular third-party plugins include Profile Manager for creating custom profile fields, Tidypics photo gallery plugin, and the Chat plugin for adding instant messaging functionality.

All users on *Elgg* get a profile page and a landing page, known as the dashboard. Both areas can be populated with widgets that pull data from various sources to show the user's activity. You, as the administrator, can set up a default layout for all users that the individual users can then rearrange as per their preferences and requirements. Users can write blogs, create and participate in groups, and can also host their own pages, upload all kinds of multimedia content and link them along with the text content. *Elgg* also has an impressive access permission system that lets you define parameters for accessing a particular type of content.

Create a simple-to-administer network with all the features you'd need.

★★★★☆

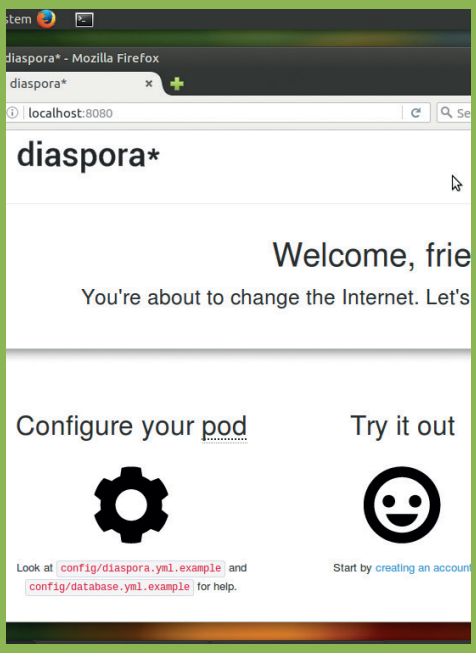
Diaspora: the distributed social network

Unity in diversity.

Diaspora is a unique social network. It works and behaves like the popular ones and lets you add friends and follow hashtags and displays updates in your stream. As a social network, Diaspora is very capable and some even credit its Aspects feature as the inspiration for Google+'s circle.

However, the biggest advantage of Diaspora is that it's open source and federated, which liberates it from the control of any one organisation. The Diaspora social network is made up of a number of servers called pods that connect to each other. The pods are independently operated, and anyone can set one up and connect it to the Diaspora network. Pods can be private to a particular group, or open and allow anyone to join – so to join the Diaspora you can get an account with a publically listed pod. Alternatively, you can also set up a Diaspora pod for your organisation.

A couple of years after crowdfunding the development, the Diaspora developers passed on the development baton to the community under the guidance of the Free Software Support Network. But this had no effect on the hosting of the network, since the actual hosting of the pods is done by individuals around the world. To join Diaspora you can sign up with any of the public pods listed at **http://podupti.me**. For more information about Diaspora, read our FAQ on this social network from Issue 8 (**https://www.linuxvoice.com/faq-diaspora**).



Follow the detailed installation guide on *Diaspora*'s website (**https://wiki.diasporafoundation.org/Installation**) to easily set up your own pod.

BuddyPress vs Community Builder

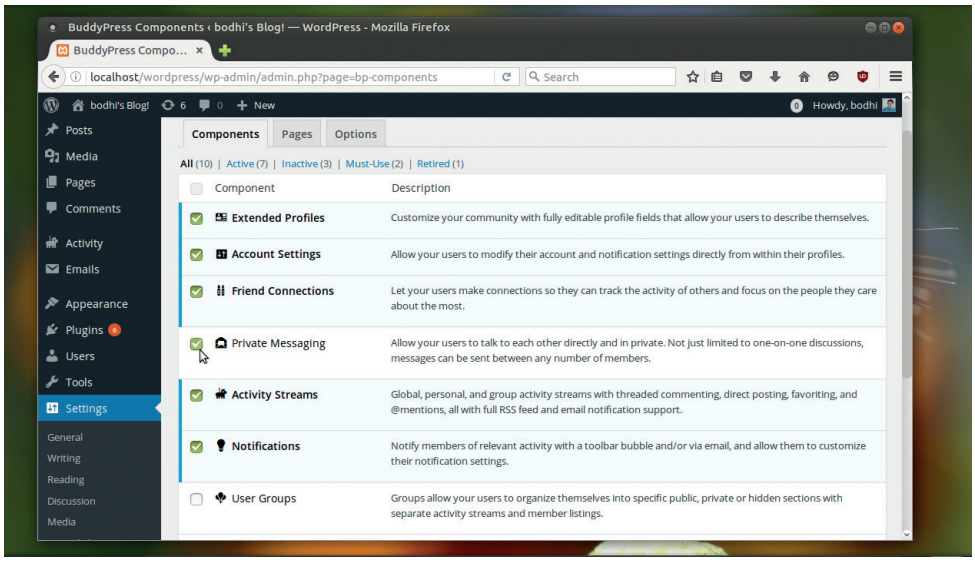
Battle of the extensions.

WordPress and *Joomla* are two very popular content management systems. Both have elaborate extensions mechanisms for fleshing out the base installation. *BuddyPress* and *Community Builder* are two different plugins that equip social networking features to *WordPress* and *Joomla* respectively.

BuddyPress is developed by Automattic, which makes it the official extension to turn a simple *BuddyPress* blog into a social network. The extension adds core networking functionality such as the ability to create profiles, add friends, create groups and message people.

BuddyPress's configuration options are housed under the Settings section in the *WordPress* administration panel. Several configuration options for *BuddyPress* also extend existing *WordPress* configuration settings: for example, the Users settings in the *WordPress* administration screen gets a new section called profile fields that you can use to let users share all types of information on their profiles.

The plugin also associates certain *WordPress* pages with *BuddyPress* pages. This enables you to create custom registration and activation pages. If you're installing it on a live *WordPress* website with registered users you'll appreciate the fact that the plugin automatically syncs *WordPress* user profiles with *BuddyPress* profiles. *BuddyPress* enables members on the website to communicate among themselves. There's also a useful notifications system for everything from friend requests to whenever someone tags you.



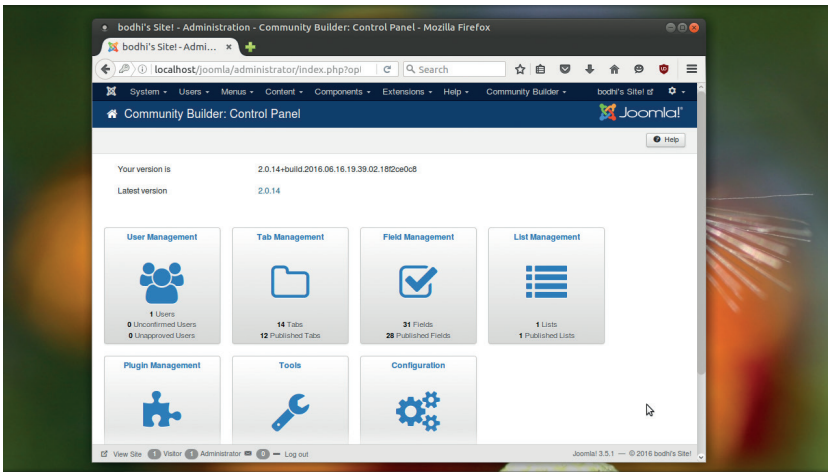
***BuddyPress* is designed to be modular. So while it includes the core networking features, for other social functionality, you'll have to install other plugins such as *bbPress* for forums.**

Foster a community

If your website is powered by *Joomla*, you can use the *Community Builder* plugin to create a social network for your users. Installation is again pretty straightforward, and once activated the plugin shows up as an additional drop-down menu in the *Joomla* administration panel. Some settings are also available from under the *Community Builder* menu in the *Joomla* components menu. The plugin draws a handful of its own menus that can be controlled from under the existing *Joomla* menus section.

On the front-end you get a *CB* login form in addition to the *Joomla* login form. Unlike *BuddyPress* however, which takes over the registration process from *WordPress* after installation, you'll have to manually turn off the *Joomla* registration form else your website will have two registration boxes.

CB lets you customise the look and feel of the user's profiles through fields, tabs, add-ons, and more. The plugin make things easier to find by organising multiple profile data fields into tabs. You can also create custom profile fields that can be of any of the 23 built-in field types including text, checkbox, image, video, file, etc. There's also a tool that'll sync the *Joomla* user table with the *CB* user table. You can get more community features via plugins.



Head to **http://demo.cbdemosites.com** to get to grips with the administration of a *Community Builder*-infused *Joomla* installation.

BuddyPress
The official mechanism for helping users socialise on a wordpress website.

★★★★☆

Community Builder
The *BuddyPress* alternative for *Joomla*-powered installations.

★★★★☆

OUR VERDICT

Create your own social network

Community Builder and *BuddyPress* aren't standalone options for creating social networks. Rather they are both designed to extend networking features to members on their respective base websites without setting another one up from scratch. Both fit snugly into the back-end of their respective content management system. installation of both is simple, so if you've got a *WordPress* or a *Joomla* website setup, you don't really need to consider any of the other solutions on offer.

The other four solutions are standalone options and can be better compared to each other. Of these, *Mahara* will appeal to the least number of users since it's designed with a very particular use case in mind. *Mahara* excels at creating a network that enables your users to showcase their achievements and engage with others based on this. Forums and groups are *Mahara*'s two main networking tools. Its administrative interface is logically divided into tabs, and you can have it up and running in no time.

Then there's *Trident*, which is young in terms of stable releases but comes from developers who have a strong pedigree of creating social networking software. Despite being available for no cost, the software isn't of much use if you don't use the keys handed out freely by its corporate overlord. If you don't have any issues with this, you'll find *Trident* easy to customise and flesh out. The Dashboard is very user-friendly and the Studio offers a unique mechanism to shape the look and feel of the installation.

Our winner

This leaves us with *Elgg* and *Oxwall*, both of which trump the others for offering the most functions and flexibility. They both ship with a good number of plugins and offer many more, including several paid ones, on their respective websites. Straight out of the box, *Oxwall* has a more appealing admin interface, which pips it over *Elgg*, which by default creates a rather uninteresting-looking landing page.

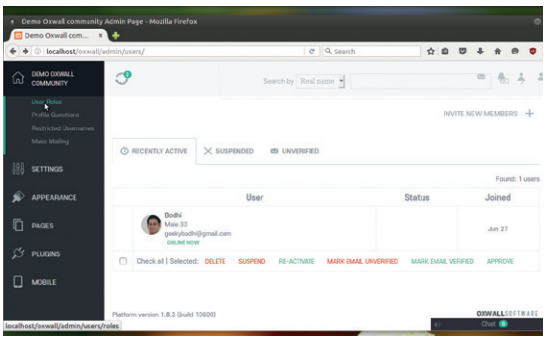
Elgg has a Twitter-like service and you can control its character-limitation. *Oxwall*, on the other hand, lets you edit its interface for mobile devices as well. Once you've set up an *Oxwall*-powered website you can either open it up to public, or restrict access to it behind a walled garden.

> Elgg and Oxwall both ship with a good number of plugins, and offer many more on their respective websites

Ready to deploy

Once you've decided to roll out your custom social network it'll be a good idea to leverage the expertise of others. Depending on the scale and size of your network you can deploy them with ready-made appliances as well as on specially-tuned web hosts. Virtually all projects including Elgg, Oxwall and BuddyPress collaborate with multiple web hosting providers and offer quick deployment and assistance.

Another deployment option is in the form of prefabricated appliances. The Turnkey Linux and Bitnami projects are two popular examples of these. Turnkey appliances are self-contained systems that run atop Just enough Operating System (JeOS) components that are required to power that particular app. The project currently produces such an appliance for Elgg. Bitnami on the other hand produces self-contained applications that include all the libraries and runtimes. While the project currently supports none of the social networking platforms covered in this group test, Elgg and Oxwall are in its upcoming list. Furthermore, you can use its LAMP stack as the base on top of which you can roll out all of the mentioned web apps.



Oxwall lets you customise all aspects of your social network including the layout and appearance without dabbling with HTML or PHP.

1 Oxwall

Killer feature Easy to administer
URL http://developers.oxwall.com
Create a social network that's intuitive to administrate and flesh out.

2 Elgg

Killer feature Twitter-like service
URL www.elgg.org
Useful for creating a user-centric social network.

3 Trident

Killer feature Studio
URL: www.online.me
Offers a unique network editor but is too dependant on its corporate sponsor.

4 Mahara

Killer feature Digital portfolios
URL www.mahara.org
Show off your work by creating a merit-based network.

5 BuddyPress

Killer feature Impressive profiles
URL https://buddypress.org
The best mechanism to add social networking function to a WordPress *blog.*

6 Community Builder

Killer feature Plugins
URL www.joomlapolis.com/community-builder
An extensive plugin to transform a Joomla *installation into a social network.*

Subscribe

shop.linuxvoice.com

Introducing Linux Voice, the magazine that:

- **Gives 50% of its profits back to Free Software**
- **Licenses its content CC-BY-SA within 9 months**

12-month subs prices

UK – £55
Europe – £85
US/Canada – £95
ROW – £99

7-month subs prices

UK – £38
Europe – £53
US/Canada – £57
ROW – £60

DIGITAL SUBSCRIPTION ONLY £38

Get 114 pages of tutorials, features, interviews and reviews every month

Access our rapidly growing back-issues archive – all DRM-free and ready to download

Save money on the shop price and get each issue delivered to your door

Payment is in Pounds Sterling. 12-month subscribers will receive 12 issues of Linux Voice a year. 7-month subscribers will receive 7 issue of Linux Voice. If you are dissatisfied in any way you can write to us to cancel your subscription at subscriptions@linuxvoice.com and we will refund you for all unmailed issues.

NEXT MONTH IN LINUX VOICE

SECURITY

Learn the essential steps that you should be taking to stop bored kids messing with your Linux machines.

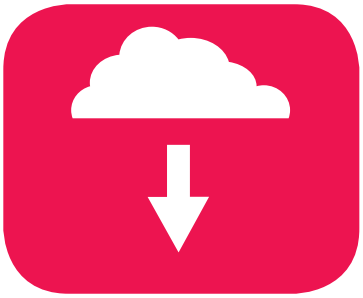
EVEN MORE AWESOME!

Ubuntu Snappy

Snappy packages – they're the hot new thing from Canonical that's going to end dependency hell forever. At least, that's the theory...

Linux everywhere

In one shape or form, Linux is on the ISS, powering the SpaceX Falcon 9 rocket and running servers in the frozen wastes of Antarctica.



Cloud storage

If you fear the mega-corporations snooping on your precous collection of cat videos, why not host it yourself with a bespoke cloud storage server?

LINUX VOICE IS BROUGHT TO YOU BY

Editor Ben Everard
ben@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Editor in hiding Graham Morrison
graham@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Creative director Stacey Black
stacey@linuxvoice.com

Editorial consultant Nick Veitch
nick@linuxvoice.com

All code printed in this magazine is licensed under the GNU GPLv3

Printed in the UK by
Acorn Web Offset Ltd

Disclaimer We accept no liability for any loss of data or damage to your hardware through the use of advice in this magazine. Experiment with Linux at your own risk!
Distributed by Marketforce (UK) Ltd, 2nd Floor, 5 Churchill Place, Canary Wharf, London, E14 5HU
Tel: +44 (0) 20 3148 3300

Circulation Marketing by Intermedia Brand Marketing Ltd, registered office North Quay House, Sutton Harbour, Plymouth PL4 0RA
Tel: 01737 852166

ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**
subscriptions@linuxvoice.com

FOSSpicks

Sparkling gems and new releases from the world of Free and Open Source Software

Our benevolent editorial overlord **Graham Morrison** tears himself away from updating Arch Linux to search for the best new free software.

Software planetarium

Stellarium 0.15.0

Stellarium has always been our favourite Free Software astronomy application. It's also hugely successful, available for Windows, OS X and even Android. But the thing that really grips you about *Stellarium*, and what we'd argue has made it so popular, is that even from the first screen you see, it always looks gorgeous. The default view launches full-screen (press F11 for windowed mode) and places you in the middle of a photorealistic meadow. The ambient light within this scene will be similar to the light outside your window, which means if it's daytime, the only stellar objects you'll see are the sun and moon. Press Alt and '+' on your keyboard to advance time an hour and watch how the horizon darkens through twilight and the encumbrance of the heavens present themselves.

Press + alone, and the view will skip ahead 24 hours – a brilliant way of watching the phases of the moon, or the orbits of Venus and Mercury playing with the sun. Click on any object to see detailed real-time statistics on your selection, and zoom in enough you'll even see the objects move as you watch, including phases of the moon, the angles of planets, solar eclipses and even shooting stars.

1 Object details When an object is selected, real-time in-depth statistics and details appear here.
2 Sky art The constellation art of lots of different cultures are embedded within *Stellarium*.
3 Shooting stars Stars twinkle, deep space objects are newly textured, and both atmospheric fog and light pollution help render either a realistic or a Spielberg-esque view of the sky. **4 Ocular views** Limit your view to that of your telescope or camera, for accurate predictions. **5 Backgrounds** Import your own garden, or enable one of the new 3D environments for terrestrial viewing. **6 Starlore** Alongside the excellent User Guide, there's also excellent background info on sky art cultures and other viewable elements. **7 Toolbox** Easily turn on and off viewable elements, include satellites and labels. **8 Configuration** Place your viewer anywhere on the globe, or even in space, at any time.

Distant stars

This releases uses *Qt 5.6*, which ran faultlessly even on our modest Intel GPU. This is important because almost every element within the main user interface is accelerated through your graphics hardware, often generating 60 frames per second even when updating the details on stellar bodies in real time.

The quality is good enough for projection in planetariums, and a new addition is a plugin for remote control use like this, where a guide can take an audience on a virtual tour. Our second favourite feature is the additional sky cultures, now including sky art for Macedonian, Ojibwe and Dakota/Lakota/Nakota constellations, projected into the sky. But behind this graphical stardust, *Stellarium* is also a serious tool capable of helping study and observations. There's initial support for the Washington Double Star Catalog, for example, and you can now create bookmarks for places, plus there are new functions in the wonderful scripting engine and additions to the powerful *AstroCalc* utility, brilliant for planning an outside viewing session. Plus, the User Guide has been completely overhauled. *Stellarium* is one of those brilliant examples of what open source can do. It's great.

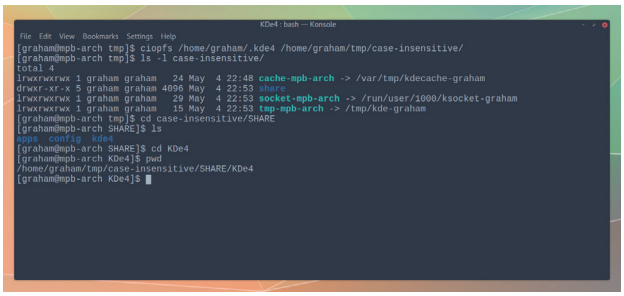
Project website
www.stellarium.org

Filesystem

ciopfs

We're used to the idea that if you don't know what you're doing, you shouldn't play with the filesystem. But there's one exception to this, and it's thanks to the wonderful FUSE (filesystem(s) in userspace). FUSE lets you – an ordinary user – create, mount and play with filesystems on your own, without messing up the integrity of the Linux system you're running on.

FUSE is great for mounting experimental write-enabled Apple or NTFS partitions, for instance, as well as experimental or development filesystems. Which is definitely the category that ciopfs falls in to. Ciopfs is old (the last update was from 2011), but we've already found a great use for it. It's a FUSE filesystem that mounts your own filesystem within itself, only the files and folders within ciopfs are no longer case-sensitive.



IF CAPITAL LETTERS IN FILE PATHS CAUSE YOU PROBLEMS INSTALL CIOPFS.

The same files' or folders' destination can be addressed as **games/SystemShock2**, **GAMES/SYSTEMSHOCK2** or **games/systemshock2**, for example. And we've just given a great big hint at how this can be used: cheap *Wine*/Windows games conversions, where the case of paths within the executable are wrong and yet hard-coded.

An easy fix for lazy paths

Pointing *Wine* at a game installed within a Ciopfs mount point will avoid any conflict, as both types of file paths will work. It's a brilliantly simple idea, and almost as simple to use. With both FUSE and Ciopfs installed, type **ciopfs source destination**, just like mounting any other filesystem, and you'll be able to freely navigation within the destination mount point either lowercase paths, uppercase paths, or a mixture of both. You can then supply this path to any problematic application having trouble with its capitalisation.

> FUSE is great for mounting experimental filesystems – such as Ciopfs

Project website
http://www.brain-dump.org/projects/ciopfs

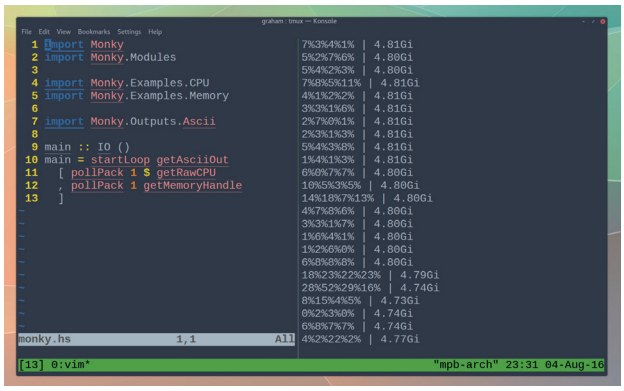
CLI status bar

Monky 2.0

Tmux, the tool that enables you to run multiple consoles alongside one another, has become the standard way we interact with most of our Linux systems, and one of its best features is the plugin system. This enables you to easily install modifications to your environment without even restarting *Tmux*, adding the ability to suspend and resume, for example.

One of the best plugins turns the bottom line of your console into a status bar, where you can add any details you choose with other plugins, such as *Git* branches, the date or unread emails, and if you have the skill, building your own plugins to display and do exactly what you need is one of of the most unexpedidly powerful aspects of *Tmux*. *Monky* does a similar job, only without the *Tmux* and console requirement, but with similarly complex configuration potential.

What makes *Monky* particularly powerful is that it uses Haskell for its configuration file, effectively turning what are normally static parameters into dynamic functions you can embed into your own status display. Fortunately, there are plenty of examples and enough modules to play with that you don't have to understand too much Haskell, but getting a decent configuration is still a challenge, even when a little copy and paste and a few parameter edits are usually all that are required. This update has changed most modules, and as such, older examples will need some modification to work, or you can start afresh. But using the modules to display details on disk usage, network usage, wifi strength and CPU usage to build your perfect status utility is a project worthy of any elite hacker's time.



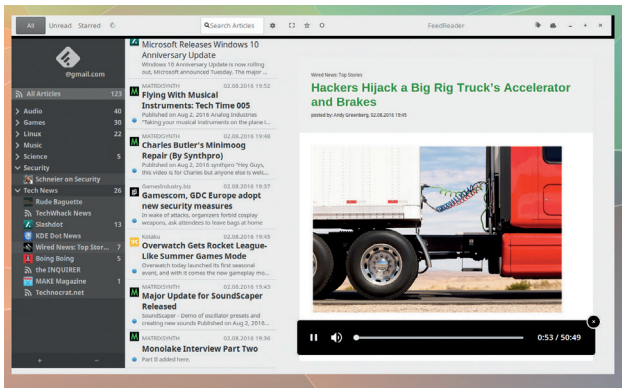
Monky **is a little like** ***Conky*** **without the dependency on X.org. It's also difficult to get running.**

> What makes Monky so powerful is that it uses Haskell for its config file

Project website
https://github.com/monky-hs/monky

RSS reader

FeedReader 1.6 beta



Thanks to its web-based UI, *FeedReader* gets very close to approximating the functionality of ye olde Google Reader.

Like many, we're still mourning the loss of Google Reader, despite it being three years since its demise. Google Reader was one of Google's best web apps, pooling whatever RSS feeds you subscribed to into a single place, and it's almost impossible to write anything about RSS aggregation without mentioning it. It was the best way of keeping abreast of 100s of stories, and nothing has come along to replace it.

While a few services, like Feedly, have started to compete, nothing really gets close to that old functionality. Offline applications were always an important part of the same ecosystem, and since Google Reader, many seem to have gone cold.

FeedReader is an exception though, and is one of the best tools for recreating a good online interface from the comfort of your own desktop. Its web-centric interface links to popular services like Feedly, and even OwnCloud, and presents your stories in the mixed hierarchy of your own categories and sub-categories just like Google Reader.

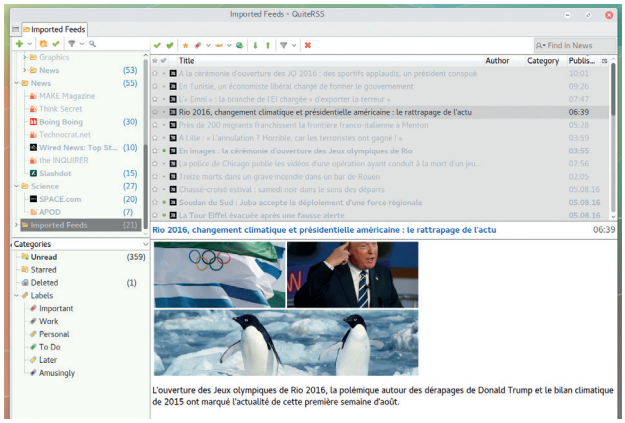
The application is great for in-place readability, but we really like its ability to push stories you're interested in to *Pocket*, perhaps for later reading on the train journey home, as well as *Instapaper* and *Readability* for non-web formatting. There's also a daemon that runs silently in the background, updating stories and occasionally notifying you of their status. This is something the web apps can't do well, and helps if you're serious about your news feeds. We typically navigate 600 stories a day, and we found *FeedReader* more than capable of helping us navigate the mess of updates (mess should be the collective noun for news).

> FeedReader's interface links to popular services like Feedly and even OwnCloud

Project website
http://jangernert.github.io/FeedReader

RSS reader

QuiteRSS 0.18.5



Penguins, the Olympic games and Donald Trump – all will be old news by the time you read this.

And still RSS won't die. Despite the social networking revolutions supposedly replacing the need for syndicated content from boring old websites, many of us depend on RSS to get updates on sites we trust, outside of their own need to 'push content'. This is what makes *FeedReader* (above) so interesting, and why there are still other applications being developed, such as *QuiteRSS*.

Almost at the opposite end of the scale to *FeedReader*, *QuiteRSS* is a great little utility recommended to us by podcast listener zmoylan. It features no background daemon, or integration with cloud services, and it can't push news stories back to *Pocket*. But it's very fast (not just 'quite' fast as its name implies), and does everything you need perfectly. We imported our feed list from an exported OPML file spat out by our web client, and *QuiteRSS* read this, updated stories and populated the various categories quicker than the online equivalent, without any noticeable hit on system performance.

This is the news

Reading quality didn't suffer either, as the stories themselves are rendered using an embedded *WebKit* core browser, with layout identical to the online versions. Adblock is even included. It's also quick and easy to filter and search through your stories, and we liked the way you can switch the story view into 'newspaper' mode.

Each story in your selected category is listed in a single page one after the other, so you can scroll through them quickly to see if any catch your interest. You can quickly disable images, which is useful if you're tethered to your phone's data allocation, and stories, feeds and categories can all be opened in separate tabs, like a finely tuned web browser.

Project website
https://quiterss.org

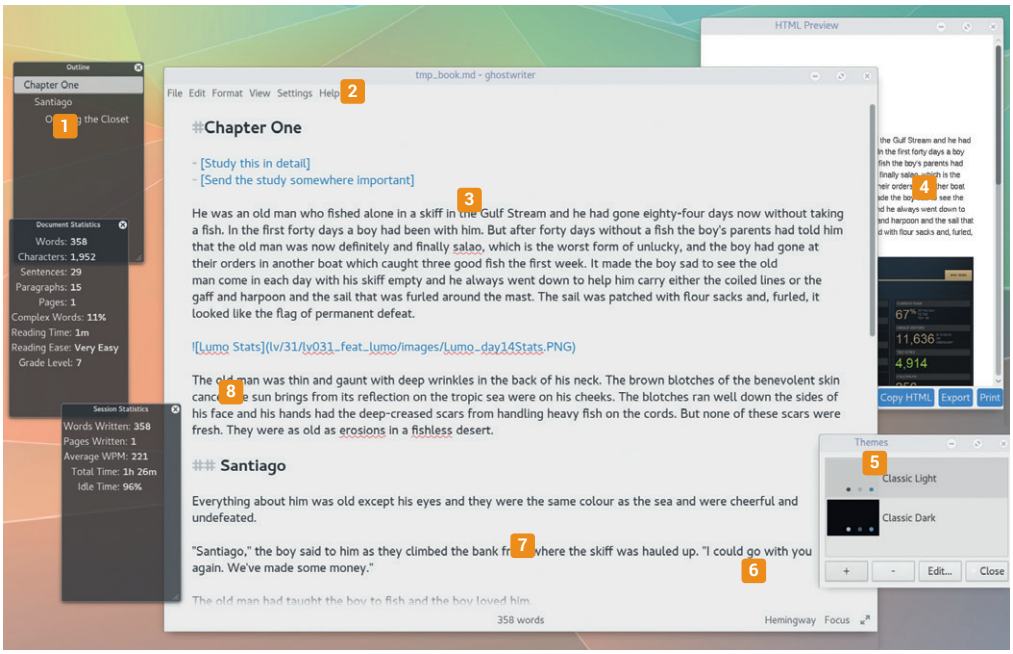
Text editor

GhostWriter 1.4.0

We obviously do a fair bit of writing. And for us, at least, there is no one-size-fits-all text editor for the various things we do. Some writing is done in *Vim*, for example, especially when working with GitHub's Markdown and writing to a specification. Other writing is done in *FocusWriter*, which is currently our favourite distraction-free editor for writing more descriptive text.

GhostWriter, as its name implies, is designed more for creative writing rather than documenting APIs or writing tutorials, but it still has a unique take on this that makes it worth a look. What makes *GhostWriter* different from something like *FocusWriter* is that it uses Markdown within the main editing window so you can add things like titles, subheadings and lists without reverting to menus or keyboard shortcuts, and without needing a special file format. As we've mentioned in previous issues, Markdown is a simple syntax for marking bits of your text document – using # for titles, for instance, only without all the hassle of opening and closing brackets, as with HTML or XML.

We really like *GhostWriter*. While we understand the overall ethos of 'distraction free' – presenting nothing but a typing-and-words interface to the writer – our practical needs outweigh our creative needs, and it's good to see *GhostWriter* starting with things like the menu being visible, as well as indicators for editing mode and word count. It's amazing how important word count is to most writers, and yet it's often almost impossible to find an editor that makes this visible all the time without eating great swathes of screen. Even with *FinalWriter* you have to drag the cursor to the bottom border of the screen, which is more of a distraction than the hopes of being distraction-free by not showing it by default.



1 Outline As you write headings and subheadings, they appear in the outline view. **2 Markdown** Using the Markdown syntax (and there's an included cheat sheet), you can mark which parts of your document are important. **3 Task List** Add things that need to be done within the text, and mark them finished when complete. **4 HTML preview** See how your writing will look after being exported to a web page. **5 Themes** Light and dark themes are included, and you have full font control. **6 Hemingway mode** Disable the Delete key for writing streams of consciousness. **7 Word count** Simple but essential when you write words for print.

Apart from the menu, there are three things you can click on in the writing window – an icon to make the application full-screen or windowed, an icon to highlight only the line you're writing on and an icon to enable Hemingway mode. Hemingway mode? This is a mode we've seen in a few distraction-free editors over the last few years, and it disables to the Delete key. This may work for some writers perhaps too troubled with perfection, but we find the Delete key useful.

> Like Hemingway, the features in GhostWriter are few but expertly chosen

Perhaps more like the real Hemingway, the other features in *GhostWriter* are few but perfectly chosen. In particular, there's an outline window that shows heading and subheadings, a document statistics overview for when you need more details, and settings for how much or how little of your writing is highlighted.

Pandoc, which we looked at last month, is used for the output, which means everything from HTML to PDF looks fantastic. Font and UI rendering is excellent too, especially on high DPI displays and a desktop with *Qt 5.6* or higher. This may seem superficial, but readability and crispness of text is one of our main reasons for using a high-DPI display in the first place, and it's a pleasure to type things into *GhostWriter*.

This may also be subjective, but it's quick too – characters appear almost instantly (as they do in *FocusWriter*), which is something you only really notice when you type lots of things. The only thing missing from *FocusWriter* we can think of is the typing sound.

Project website
http://wereturtle.github.io/ghostwriter

PulseAudio mixer

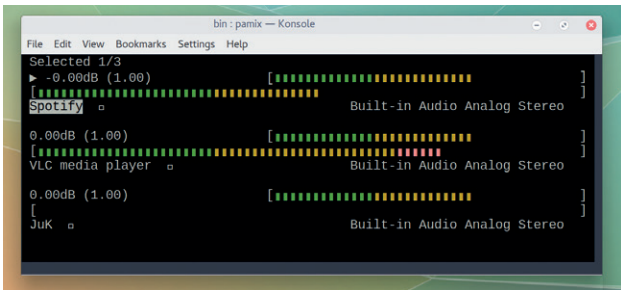
PAmix

There are almost as many software audio mixers as there are audio subsystems for Linux. There are even a few for the command line. We've always relied on *alsa-mixer*, for example, when our machines are making a sound we just can't stop any other way.

PAmix is an *ncurses*-based mixer for *PulseAudio* that has no dependencies (other than *ncurses* and *PulseAudio*) and gives you instant access to the audio output on your hardware. It's not powerful and it's very simple in its functionality, but it's also tiny and runs instantly. It was quicker to build the entire project from its source code that waiting for the graphical *pavucontrol* mixer for *PulseAudio* to load on the desktop, for example, and it provides just enough functionality for the majority of users.

Each application that's sending audio to *PulseAudio* is listed vertically, and alongside each of these there's a volume slider. Each audio source also has a bouncing volume indicator, just like the VU meter on an 80s stacking Hi-Fi, and it's very quick and easy to see what's responsible for whichever sound you want to control.

Pressing M will mute or unmute the audio source, and you can control the volume with the H and I keys. J and K are used to switch between inputs and Q to quit. That's almost all there is to this small tool, but it's about all you need 90% of the time when dealing with normal audio issues, and the clarity of the output and control is better than you find in desktops like KDE, for instance, although this has improved drastically over the last six months. If you ever need more control, there's always *pavucontrol* and even *PulseAudio* on the command line, but if you want quick visualisation over a remote Raspberry Pi audio client, or you run everything from the terminal, *PAmix* is an excellent addition.



The build pamix executable is a mere 56k in size.

> PAmix provides all you need 90% of the time when dealing with audio issues

Project website
https://github.com/patroclos/PAmix

Games controller config

SC Controller

Valve's Steam Controller has now been out for almost a year, and the verdict is still undecided. It does away with the traditional dual analogue sticks, and replaces them with circular touchpads and clever haptic feedback, the intention being that they can act as sticks if you want them to, but they can also act more like a mouse, which is essential for FPS games, for example. We use ours alongside a regular controller for playing games such as *Kerbal Space Program* – a game that otherwise requires a mouse and keyboard.

Steam Controllers work with Valve's Linux-based Steam Boxes, and SteamOS, where the community have created profiles for most games, but they've also been difficult to use outside of Valve's ecosystem. Until now!

SC Controller is a brilliant utility that enables you to create profiles and reconfigure every aspect of the Steam Controller, without ever having to launch Steam. This is particularly useful if you want to use the controller on a computer that doesn't have Steam installed, or a computer where you don't want to install Steam.

It supports the redefinition of the single analogue stick, the pads and the orientation input, as well as emulation of other controllers such as a real 360 controller, a mouse, a trackball and a regular keyboard. You also have control over pressure thresholds, analogue output (great for mouse control) and even the ability to trigger scripts and commands. It takes Steam Controller out of the world of niche gaming and into a world of huge potential, especially if you can pick up a Steam Controller in one of Valve's famous seasonal sales.

Turn your Steam Controller into a keyboard and mouse, all without Steam (thanks Ioangogo!).

> SC Controller enables you to reconfigure every aspect of the Steam Controller

Project website
https://github.com/kozec/sc-controller

FOSSpicks Brain Relaxers

Space adventure

Oolite 1.84

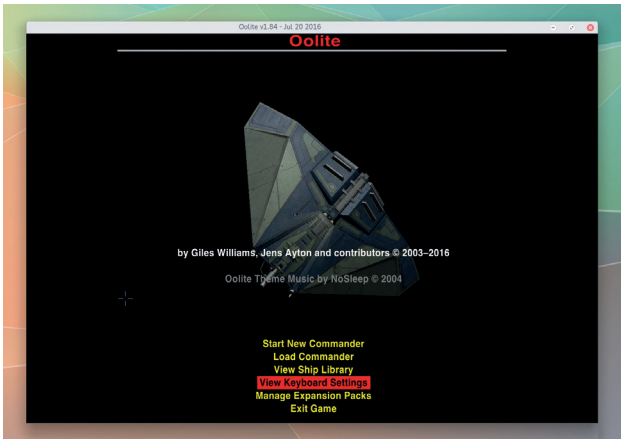
We've of course covered *Oolite* before. It's absolutely one of the best open source games you can play, and has been the best interpretation of the 1980s classic, *Elite*, for over a decade. *Elite*, for the uninitiated, was a game set in the vastness of space. You could be a trader, a pirate, a mercenary, or a mixture of all three.

The 3D graphics were also revolutionary, helping pull the player into a world that felt infinite. David Braben, one of the original developers of *Elite*, is now head of Frontier Developments, a games company that has spent the last couple of years developing *Elite Dangerous*, though sadly there's no Linux version. This makes *Oolite* a genuine alternative, although it's much more like *Elite* than *Elite Dangerous*, and some players will prefer it for this.

Right on, commander!

Updates come few and far between – the last one was in May 2015, but each one is a signification improvement over the previous version. In particular, the game's AI and combat have become a lot more satisfying.

Version 1.84 makes a few important changes, such as putting HUID messages under the console, and allowing all ships to carry multiple lasers, but it's never lost the attention to detail. Even the movement of the tumbling ship, for example, has been enhanced to make it more like the pseudo-random of the original BBC version, and there are many, many bugfixes too. Fundamentally, it's a brilliant game and *Oolite* is the best version.



If you don't have the latest PC (and Windows!) for running *Elite Dangerous*, *Oolite* is a brilliant alternative that's more true to the original.

Project website
http://www.oolite.org

2D platformer

ReTux

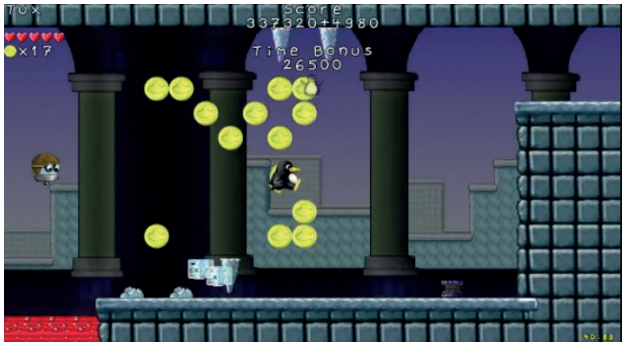
Despite being completely Free Software and proud of it – declaring itself as a "100% libre software and libre culture action platformer loosely inspired by the Mario games," this cute old-school game has taken an unusual approach to distribution. All downloads, including source code, are encrypted.

You need to pay $4.99 USD or more to receive the password that unlocks the files and gain access to either a set of executable binaries, or the full *Git* log and source code from the development repository. There isn't even a demo version you can try. But we respect the developers for trying a different funding strategy, and while anyone can presumably ask for the source code, or re-distribute it themselves, there aren't any obvious mirrors offering the same files.

La mode classique

On first glance, the game looks a lot like the old Linux platform game *SuperTux*. But *ReTux* isn't a copy. The visual similarity is because *ReTux* has taken the art assets and used them within its own game engine, implementing its own gameplay mechanics such as infinite lives, defaulting to running rather than walking, super-powers that come from special items and a colour palette that is apparently accessible to colourblind people.

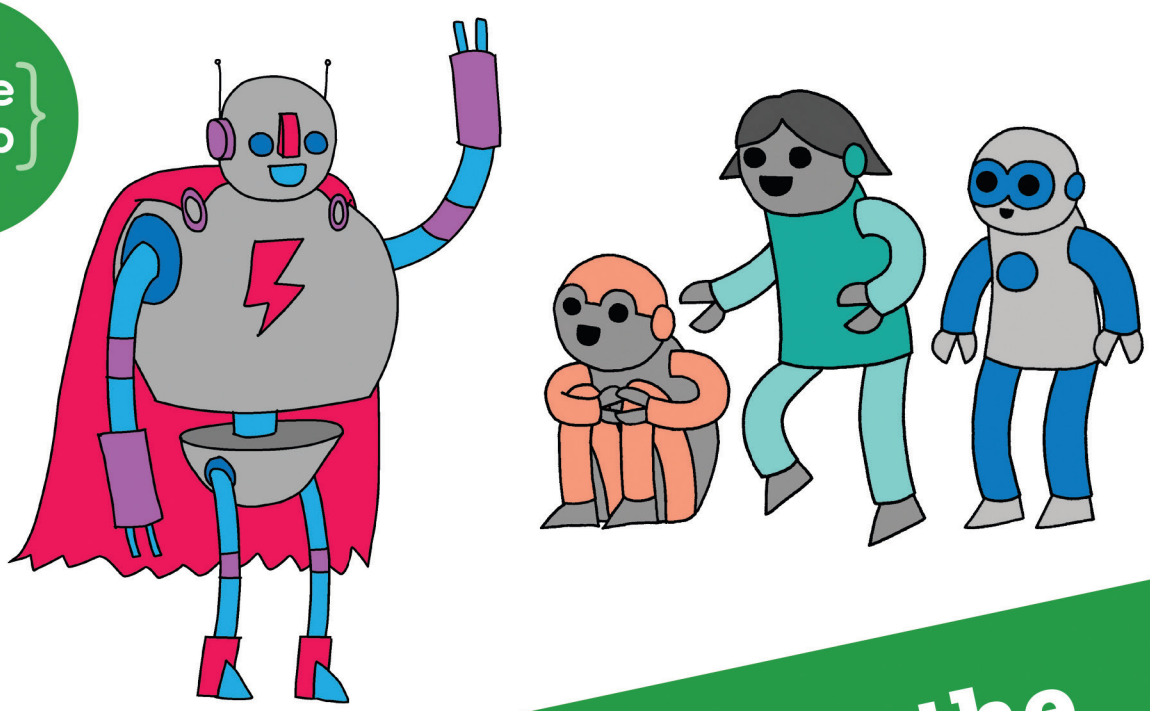
There are 41 levels, plus a level designer, with some puzzle elements. The way you progress and the way you navigate the island setting will feel very similar to *Super Mario* or *The Great Giana Sisters*, as will the coin collecting, jumping and throwing things as you attempt to reach the flagpoles at the end of each level. It's now a classic game mechanic and it's obvious a lot of thought and appreciation of the genre has gone into the game.



***ReTux* may look like *SuperTux*, but that's only because *reTux* has borrowed some of the same images.**

Project website
www.nongnu.org/retux

code club



Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

TUTORIALS

Warning: excessive Linux knowledge may lead to fun and more efficient computing.

Mike Saunders
Finds too many TLAs a PITA.

One of the terms we often use in Linux Voice is FOSS, for "Free and Open Source Software". It's a useful shorthand way of expressing a couple of philosophies in our community, but it's also a bit of a cludge. Can't we settle on one simple description? I understand that from the perspective of GNU founder Richard Stallman, the "free" highlights freedom more than low cost or practicality. But it can be misinterpreted – for instance with dodgy Windows "freeware".

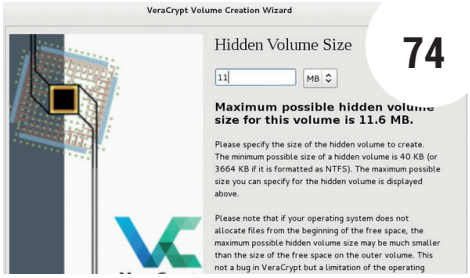
Indeed, I asked Richard about this a few years ago: whether, looking back, another word like "libre" may have made more sense. He seemed to agree that "free" has led to ambiguities, but hammering home the "free as in speech, not beer" line has helped to explain. Personally, I like to use the term "Free Software" as the freedom is incredibly important to me – the freedom to use, share and modify software. Freedom from spying governments and companies. Freedom to keep software and hardware running for as long as its users wish, without enforced obsolescence. I've nothing against the term "open source", but I do share Stallman's concerns.

mike@linuxvoice.com

In this issue . . .

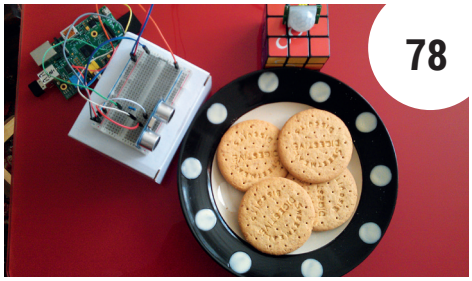
Boinc: Help scientists do science more betterer

Advance the world of science with *Boinc*, and become a scientist without even leaving your desk. **Ben Everard** explains all.



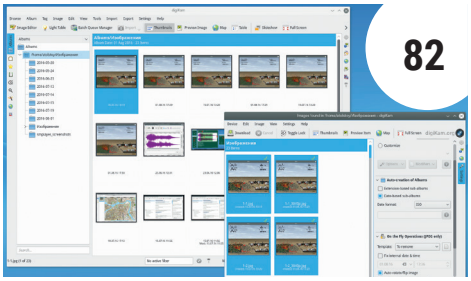
Encryption with Veracrypt

Nate Drake checks out the key features of the latest fork of *Truecrypt*, employing military-level encryption for your data.



Raspberry Pi: Physical computing

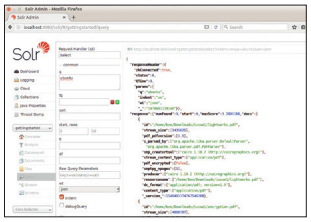
Go back to basics with **Les Pounder** and discover physical computing with Python, a Raspberry Pi, and a bunch of LEDs.



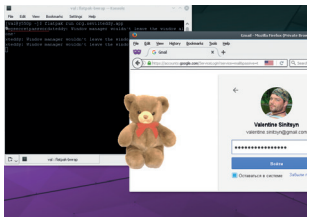
Photography: image processing

Apply smart fixing, processing and storing techniques to your images. **Alexander Tolstoy** saves you time, so you can watch more cat videos.

Coding



Solr **86**
Add some order to the chaos that is the Linux Voice back catalogue with a search engine.



Build packages **90**
Valentine Sinitsyn explains the tech and tricks behind creating nicely wrapped-up packages.

ADVANCE THE WORLD OF SCIENCE WITH **BOINC**

Ben Everard gets a labcoat and becomes a scientist without ever leaving his desk.

BEN EVERARD

Why do this?

- Help scientists do science
- Put spare computing power to work
- Save the polar bears

Boinc is a bit of software that lets you pass on your spare CPU time to projects that need the computing power. By dividing a task between thousands of *Boinc* users – each of whom contribute some of their spare CPU cycles – research institutions can get a huge amount of computing power to try to improve the sum of human knowledge. There are a huge range of projects on *Boinc*, including analysing data gathered by telescopes, building our understanding of diseases and investigating the structure of proteins. By working with this software, you're contributing to these highly worthwhile fields.

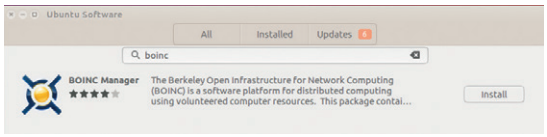
STEP BY STEP: SOLVE CLIMATE CHANGE

1 Install software

If *Boinc* isn't in your distro's repositories, you'll need to grab it from the project's website: **https://boinc.berkeley.edu**. This will download a shell script that you need to run to create the **BOINC** directory. Inside this directory you'll find two programs called **run_client** and **run_manager**. Run these with:

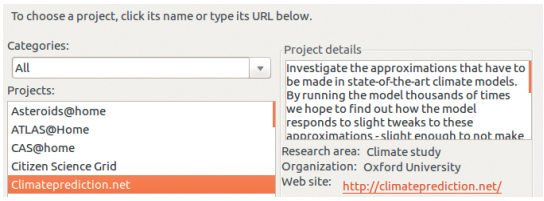
```
./run_client --daemon
./run_manager
```

There's detailed troubleshooting information at **https://boinc.berkeley.edu/wiki/Installing_BOINC**.



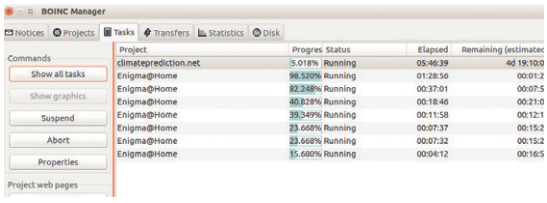
2 Select a project – Climateprediction.net

There are a huge range of Boinc projects, most run by researchers at universities around the world; we'll use the **climateprediction.net** project, as it runs on Linux and it's a project that we think is important. Open up *Boinc* and go to Tools > Add Project, search for **climateprediction.net**, and then press Next. For the first use, we recommend not joining a team or adding extra details, as you can do this later if you choose.



3 Advanced mode

Boinc's default interface is designed to make it easy for non-technical people to join projects, but we prefer the Advanced View. Go to View > Advanced View (or press Ctrl+Shift+A) to switch the interface. The Advanced View gives you a tabbed interface where you can see much more about what's going on including the specific tasks that are running, the amount of storage that's being used and statistics for how things are running. Different projects run in different ways, so it can be interesting to see how they make use of your computer.



4 Will it gobble your CPU? Nice!

All *Boinc* processes have a nice value of 19. The nice value is used to determine what processes to run on the CPU at a given time. The higher the nice value, the less likely the process is to run, and 19 is the maximum they can be. In other words, the nicer a process is, the more it lets other processes run first. If there are any other processes that need to run, the kernel will select those rather than the *Boinc* ones, so your machine's performance shouldn't suffer.

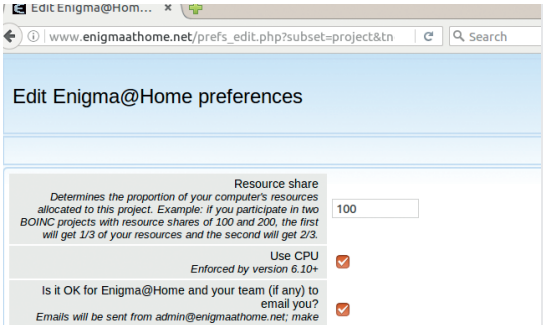
```
ben@ben-All-Series: ~/Downloads/BOINC
top - 18:51:09 up 1 day,  4:27,  2 users,  load avera
Tasks: 318 total,   9 running, 309 sleeping,   0 stop
%Cpu(s):  3.1 us,  1.6 sy, 92.0 ni,  3.0 id,  0.2 wa,
KiB Mem :  7851388 total,   464316 free,  4763496 use
KiB Swap: 16620540 total, 16528124 free,    92416 use

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU
26718 boinc     39  19     864    532    408 R 100.0
26831 boinc     39  19     864    532    408 R  99.7
26639 boinc     39  19     864    536    408 R  98.0
26469 boinc     39  19     864    532    408 R  97.7
26950 boinc     39  19     864    532    408 R  96.7
```

5 Adding more projects

You can add as many projects to *Boinc* as you like, but since you only have a set amount of computing power, the more projects you add, the less CPU time each one gets. *Boinc* will, by default, split your computing power evenly between all projects you add.

You can control the share of your computing power that each project gets in the project's settings. Highlight a project in the Projects tab and click Your Account. This will open your web browser and take you to the project's settings page where you can configure all your settings. Go to the project's Preferences link (eg ClimatePrediction.net Preferences), and click Edit. You can now increase or decrease the share of your CPU that this project gets. The higher the number, the more CPU time it gets.



6 Set up a team

Teams in *Boinc* are used to report where the computing resources have come from. They're not important in the grand scheme of things, but enable people to show how much computing resources they're contributing as a group. For example, you can see the most active teams on Climateprediction.net here: **http://climateapps2.oerc.ox.ac.uk/cpdnboinc/top_teams.php**. You can apply to join a team by searching for a team, and clicking 'Join This Team' on their webpage. Alternatively, if none of them are a good fit, you can create your own team by going to the Teams website (for climateprediciton.net, this is **http://climateapps2.oerc.ox.ac.uk/cpdnboinc/team.php**), and clicking Create A Team. Creating a team could be a good option if you have several computers that you contribute from and you want to see how powerful they are.

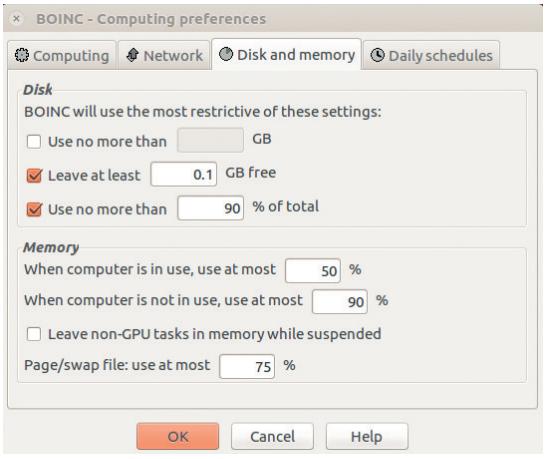
Top teams

Rank	Name	Members	Recent average credit	Total credit	Country	Type
1	gridcoin	978	410,409	184,219,446	International	Computer type
2	SETI.Germany	2952	172,851	405,080,912	Germany	None
3	UK BOINC Team	1164	136,160	163,436,574	United Kingdom	None
4	L'Alliance Francophone	4879	128,388	561,523,538	International	None
5	University of South Carolina	28	126,169	41,786,126	United States	University or department
6	USA	2881	100,913	255,341,690	United States	National
7	BOINC Synergy	929	92,360	289,694,520	International	None
8	Canada	1438	90,630	293,554,640	Canada	National
9	BOINC@AUSTRALIA	1241	80,841	241,671,099	Australia	None
10	Czech National Team	2876	80,532	381,697,293	Czech Republic	None
11	SETI.USA	956	70,534	116,918,521	International	None
12	BOINC@Finland	269	66,653	72,121,795	Finland	National
13	Russia	1246	64,714	148,704,933	Russia	National
14	PacificNorthwest	508	56,209	185,030,917	United States	Local/regional

7 Settings

In the default setup, *Boinc* should stay out of your way and just run in the background when you have spare CPU cycles. However, if you do encounter a slow-down, there are configuration options that you can use to tweak the way the software runs.

Under Options > Computing Options you can set limits on the amount of computing power that can be allocated to *Boinc* projects such as a limit on the total amount of CPU time or CPU cores. The Disk And Memory tab in this options screen is particularly useful, as these are often the resources that *Boinc* can cause problems with. Another way of limiting *Boinc* is getting it to stop when particular software is running – this is great if you have some software that requires plenty of computing horsepower that you don't want to slow down.



8 Boinc mobile

Most people have a low-power computer that runs all day every day – their smartphone. Although the computing power of a single phone is quite low when compared with a desktop, the combined power of thousands or millions is still significant. At present, the software's only available for Android, and you can find it in the Google Play Store.

The obvious concern with sharing your phone's computing power is that it will drain the battery or use up mobile data. *Boinc* on Android will only start processing if the device is connected to the mains and fully charges, and it will only transfer data over Wi-Fi. For most people, this means that it'll happily share your phone's computing power overnight, but not run when you're out during the day.

ENCRYPT YOUR FILES WITH VERACRYPT

Explore the features of the latest fork of the Truecrypt safekeeping solution to employ miltary-level encryption on your data.

NATE DRAKE

Why do this?

- Use multiple encryption ciphers simultaneously to protect files in case any one is broken.
- Use plausible deniability to keep your data safe even if you're forced to hand over a password.
- Combine passwords with keyfiles to strengthen hugely the protection of you encrypted information.

Truecrypt is dead. Long live *Veracrypt*! On 28 May 2014, the developers of the handy encryption program *Truecrypt* suddenly announced that they would no longer continue maintaining the software.

The mysterious programmers behind the project recommended switching to *Bitlocker* instead. This came as a bitter blow to those who had previously used this cross-platform utility. *Truecrypt*'s popularity stemmed from the fact that it provided a simple GUI to create encrypted containers of any size wherein files could be placed. It also enabled users to quickly and easily encrypt an entire USB stick or hard drive. Users of the Windows version also had the option to encrypt their entire operating system.

For Linux users, there were few viable alternatives that allowed such powerful encryption to be employed so easily, which is perhaps one of the reasons why French IT security consultant Mounir Idrassi decided to fork the *Truecrypt* project and create *Veracrypt*.

Version 1.17 of *Veracypt* was released in February 2016 and continues to be maintained. Although the GUI is not particularly intuitive (as with *Truecrypt*) it only takes a matter of minutes to install and start using it to protect your most sensitive personal information.

The volume creation wizard helpfully explains how each encryption algorithim is applied.

Three cheers for cascades

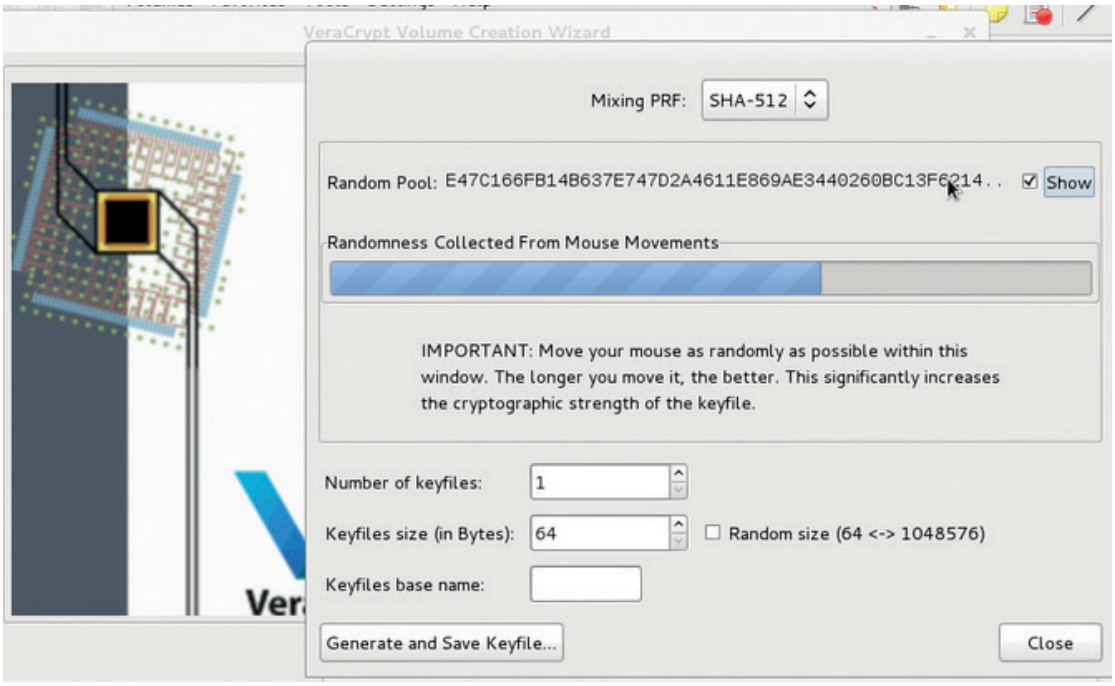
Most popular distros of Linux, including Ubuntu and Linux Mint, now offer system encryption by default as well as the option to encrypt your home folder. Gnome's disk utility has the option to format a drive with LUKS so it requires a password to access. This begs the question why there's any need for additional encryption software.

Aside from the benefits of being cross-platform, the answer is that *Veracrypt* employs cascades of ciphers. In addition to the default encryption algorithm used by LUKS, which is AES, *Veracrypt* also offers Serpent and Twofish (Twofish was developed by security guru Bruce Schneier, and is based on Bruce's previous block cipher Blowfish, which was developed in 1993 but is still very widely used). Most crucially, *Veracrypt* allows creation of encrypted file containers or drives using any one of these drives or a combination such as AES-Twofish-Serpent.

The degree to which using cascades of ciphers protects your data rather than relying on just one really can't be overstated. For instance, if you choose AES-Twofish-Serpent, each 128-bit block of data is first encrypted with Serpent, then with Twofish , and finally with AES. Each of the cascaded ciphers uses its own key. All encryption keys are mutually independent so that all three ciphers must be broken without the correct key to access your data.

There is no need to remember this however, as the volume creation wizard explains each cipher when selected in the drop-down menu as well as how each cascade of ciphers operates. Naturally employing multiple ciphers will also come at a cost in processing speed, so it's best to make use of the handy 'benchmark' button to test the speed of each algorithm in RAM.

The volume creation wizard also enables you to choose your own hash algorithim; currently the options are SHA-256, SHA-512 and Whirlpool. One of the criticisms of *Veracrypt*'s predecessor *Truecrypt* was that it used the rather dated RIPEMD-160. *Veracrypt* can open old *Truecrypt* volumes using this



Use entropy from the mouse to generate random keyfiles. You can also randomise their size.

hash, but for security reasons no longer employs it to create new volumes.

Keyfiles knowhow

If cascade ciphers and more secure hashes don't tip the balance in *Veracrypt*'s favour, the volume creation wizard also enables the use of keyfiles in addition to a password. In simplest terms a keyfile is a file whose content is combined with a password to open a volume. Without both the keyfile (or keyfiles) and the correct password, the encrypted *Veracrypt* volume will not open.

This provides a huge benefit firstly in that it will become much harder to 'brute force' your password as the encryption key will be combined with the data in the file. Secondly, it is more in keeping with the modern standard of two-factor security such as a password combined with a security token or smart card. If the keyfile itself is protected by a password, this also enables multiple people to access the same encrypted volume. The **veracrypt** volume can be created using only a keyfile to open it, and each person can then encrypt said keyfile with the password of their choice.

Veracrypt has a built-in generator for random keyfiles, using entropy gained from wiggling your mouse within its window. The downside of using *Veracrypt*'s own keyfiles is that they are quite obviously just random data, so would be easy to detect.

> Without both the keyfile and the correct password, the encrypted Veracrypt volume will not appear

One way to increase security is to generate a number of keyfiles but only use a certain number

Getting started

Head over to https://veracrypt.codeplex.com/releases/view/619351 to download the Linux version of Veracrypt. The more security-minded might also wish to verify the download via the PGP signature. The user guide can also be downloaded from this page.

Use your favourite archiving program to extract either 'veracrypt-1.17-setup-gui-x64' or 'veracrypt-1.17-setup-gui-x86' depending on your processor architecture. If in doubt, choose the second of the two.

Next in Terminal run

```
chmod a+x
```

code plus the filename to make it executable and run it. You'll be welcomed to the installer. Choose to run 'Install Veracrypt' rather than extract the Tar Package and agree to the terms and conditions. You'll also be told that to uninstall Veracrypt you'll need to run the script 'veracrypt-uninstall.sh'.

If you prefer to download and compile the source code yourself, download it from https://veracrypt.codeplex.com/wikipage?title=Downloads extract it and open README.md in your favourite text editor. Scroll down to the section titled 'Instructions for Building VeraCrypt for Linux and Mac OS X' and follow the steps there.

If you're using a GNOME Desktop Environment, veracrypt should appear right away in your Accessories post-install. Otherwise you can run it from terminal with

```
sudo veracrypt
```

Veracrypt needs admin rights to be able to mount an encrypted volume so may ask for your password if you choose to run without

```
sudo
```

The hidden volume lies inside the outer volume. *Veracrypt* will let you know the maximum possible size to avoid overwriting data.

of them, eg only the last three files of a group of 10. It's also possible to use any kind of file as a keyfile, such as an MP3 or JPEG. Provided there are enough picture/music files on your system, an adversary would have a much harder time divining which one is the keyfile. If you choose to use an ordinary file to do this, however, bear in mind that if even one of the first 1024 kilobytes of your file changes, the key will no longer be valid.

> Veracrypt has an option to create a hidden volume inside your main encrypted container or drive

Plausible deniability

Although any file can be a keyfile, it's not easy to hide the fact that you are employing encryption in the first place. Gnome's disk utility, for instance, makes no effort to hide the LUKS header if used to encrypt a drive, so while it may not be possible to access your data, it may be clear to your adversary that you have something to hide and you may run afoul of your country's key disclosure laws.

In the UK for example, the Regulation of Investigatory Powers Act (2000) requires members of the public to provide their passwords when requested to the police. Failure to comply can result in up to two years in prison. In 2010 a 19-year-old in Lancashire was sentenced to 16 weeks in a young offender's institution for refusing to cooperate with a formal order to hand over his password.

Veracrypt encrypts all volume headers, so by analysis of a drive alone, it's not possible from a technical perspective to determine that the data on it is encrypted. The data appears to be random and could be there as a result of a secure wipe of the drive by using

```
dd
```

from the Linux command line or the free erasure software *Darik's Boot and Nuke*.

While this may be true theoretically, key disclosure laws in many jurisdictions place the burden of proof upon the suspect. An unscrupulous adversary may also employ 'rubber hose cryptography' ie torture or coercion to obtain a password from a volume they suspect to be encrypted. Moreover, if you use *Veracrypt* to create an encrypted file container rather than encrypt an entire drive, there is really no plausible reason for you to have a large block of useless random data on your computer.

For this reason, *Veracrypt* has an option to create a hidden volume inside your main encrypted container or drive. As the name suggests, the hidden volume is created within the free space of another *Veracrypt* volume (the 'outer volume'). When the outer volume is mounted, it is impossible for an adversary to be certain that there is a hidden volume inside, as any free space is always hidden with seemingly random data anyway. Even the headers to the hidden volume are encrypted, so there is no way to determine if they are valid data without the correct password.

When using the volume creation wizard, you have the option to create a hidden volume with a separate password and/or keyfiles to the outer volume. Once your outer volume is mounted, copy some plausible-looking files there and make a note of the password. This is the password that you would give up if compelled to provide one, which would lead your adversary only to the dummy files. The hidden volume would be mounted separately with a different password, and would contain your truly personal files.

When the outer volume alone is mounted, even *Veracrypt* cannot determine if a hidden volume exists, and if you copy data to it, it may overwrite files already in your hidden volume. Fortunately *Veracrypt* has a

Volume creation

To create an encrypted volume on *Veracrypt*, head over to the Volumes menu then Create New Volume.

You'll first be asked to choose whether to create a file container which can be mounted as a virtual drive and have files placed inside it, or to encrypt an entire non-system partition such as a USB stick. Choose accordingly and then click 'Next'. Choose your volume type, in this case, Standard. You'll next be asked to choose the location of your file container, so click on Select File to choose a location and a name eg **myfiles.vc**. Clicking Next will let you choose your encryption algorithim and hash. If you're stuck, stay with the default options (AES + SHA-512), then click on Next.

You'll be asked to specify the volume size. *Veracrypt* will also state how much free space is available. Click on Next to choose a password and add any keyfiles. Once these are entered, click Next to choose a filesystem. The default of FAT is probably best provided you don't plan on storing files larger than 4GB. (You can choose Linux formats and even NTFS if you wish though.) Finally you'll be asked to move your mouse inside the *Veracrypt* window to gather randomness. The longer you do this, the harder it will be to brute-force your volume password.

Plausible deniability

Although it can be difficult to convince a shadowy government organisation that your encrypted files are really just random data, the first step is to use an external drive, rather than create a file container. The *Veracrypt* project recommends drives that don't implement wear-levelling such as SSD's and USB sticks, as traces of a hidden volume may remain, so consider using an old-school hard drive with magnetic storage if available.

Once inserted, head to the Volumes menu on *Veracrypt* and Create New Volume. Next choose to 'Create a volume within a partition/drive'. Click on Next then on Hidden Veracrypt Volume. Click Next then Select Device, then the main partition of your external drive eg **/dev/sdb1**.

You'll be asked to choose the outer volume encryption algorithim and hash. You can then specify an outer volume password and/or keyfiles. Remember these must be plausible, so choose a secure password (20+ characters). Click Next to start generating randomness from your mouse movements, then Format to begin creating the volume. Any files already on the volume will be lost.

Once volume creation is complete, you'll see it's been automatically mounted. Click on Open Outer Volume and copy some plausible-looking files into it. Click on Next when you're done to begin creation of the hidden volume. Choose whatever Cipher and Hash Algorithm you wish, then click next. You'll then choose the size of the hidden volume. *Veracrypt* will tell you the maximum possible size. Choose Next and then Format to create the hidden volume.

It's highly advisable to visit *Veracrypt*'s wiki and read Security Requirements and Precautions Pertaining to Hidden Volumes for some help on best practices for plausible deniability.

feature named 'Hidden Volume Protection' To prevent this when mounting the outer volume on *Veracrypt*, click Mount Options when accessing the outer volume and enter the password there to your hidden container. In this way, the data inside your hidden volume will not be affected.

Plausible deniability will naturally only be effective if the data in the outer volume is something you would genuinely wish to hide, so placing vintage episodes of the TV show *Blockbusters* in there, for example will be unlikely to impress an adversary.

Veracrypt viability

No security solution is perfect. While accessing an encrypted volume in *Veracrypt* for instance, the operating system may write to unencrypted volumes. The existence of a hidden volume may be revealed by analysing an encrypted drive or container at different stages eg if you place it inside a cloud storage service like Dropbox. The *Veracrypt* website recommends storing encrypted data offline and accessing via a live Linux CD where possible. Some of the wariness of *Veracrypt* stems from the *Truecrypt* project. In April 2015 a full security audit of the 70,000 lines of code in *Truecrypt* was completed by the Open Crypto Alliance Project. The full results of their findings are available from the Alliance's website, but in brief no evidence of any deliberately coded backdoor was found, nor were any major vulnerabilities.

Example of a Cold Boot attack. Cooling RAM chips with compressed air preserves data in virtual memory, like such as encryption keys, which can then be harvested with specialised pen-testing software.

Source: https://www.ethicalhacker.net/features/root/using-cold-boot-attacks-forensic-techniques-penetration-tests

Many of the updates made to *Veracrypt* have been done to simplify the original *Truecrypt* code used as well as make it easier for users to download, verify and compile the source code themselves as any privacy-minded person should. Some of the improvements made by *Veracrypt* have resulted in longer access times to mount encrypted volumes, although this has been cut in half with the most recent release of the software.

Mounir was kind enough to take time to speak to Linux Voice about the *Veracrypt* project. Of late, the main focus has been on developing support for Windows UEFI encryption, as he claims that currently there is no open source software available for this. Nevertheless the Linux version is under active development, and anyone interested in finding out more or devoting their time is welcome to contact the project on veracrypt@idrix.fr.

Veracrypt also will not protect against so-called "Evil Maid" or "Cold Boot" attacks, whereby an adversary has physical access to the machine on which you mount the volumes. This vulnerability isn't unique to *Veracrypt* but can be reduced by employing keyfiles and keeping your machine physically secure.

Finally, given the extensive security features already listed, it should be obvious that failure to put in the correct password and provide all the correct and unaltered keyfiles will result in the encrypted volume failing to open. Either make sure to write these down or use a password manager like *KeePass*.

Whether you're a hardened security guru or a beginner, *Veracrypt* offers some dazzling security features. As with all security issues, it's important to explore these yourself and invent a solution that works for you.

Nate Drake is the author of "Locking Down Facebook", a free guide to protect your privacy while using the social network.

PYTHON AND PHYSICAL COMPUTING

Go back to basics and introduce physical computing using the Raspberry Pi.

LES POUNDER

Why do this?

- Learn electronics
- Learn the basics of Python

You will need

- Any model Raspberry Pi running the latest Raspbian release
- A mixture of female to male and male to male jumper wire
- 3 x LED (red, amber, green)
- A breadboard
- 3 x 330Ω resistor (orange, orange,brown)
- 3 x momentary switches (buttons)

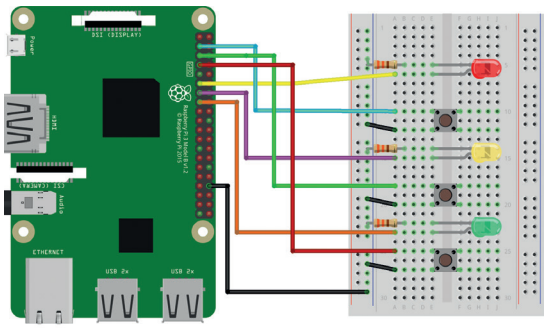
Taking your first steps into coding and physical computing can be a little daunting. Even the relatively simple task of turning on an LED can fill a new user with dread. But this challenge is immensely rewarding and will unleash a new way of thinking – the way a maker thinks! So to ease you into the new world of the maker we shall use three projects, each of which will build upon the last and introduce new programming concepts. The goal of these projects is to get ourselves familiarised with hardware components and how we can control them in different, and relatively simple ways.

For all of the projects we shall use one simple circuit. The basic circuit comprises an LED that is connected to the GPIO of the Raspberry Pi. The long leg, the anode, is connected to a specific GPIO pin; this is set in the code for each project later in this tutorial. The short leg, the cathode is connected to a ground rail on our breadboard via a 330Ω resistor. This circuit is replicated so that there are three occurences on the breadboard. A push button is also inserted into the breadboard, so that the legs of the button are across the centre channel of the breadboard. The button is also connected to a specific GPIO pin, and to the common ground rail of the breadboard.

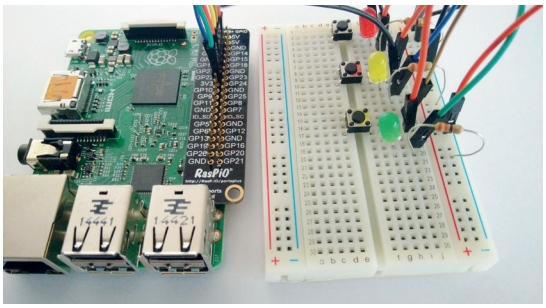
Please refer to the diagram at **https://github.com/lesp/LV31-Python-Basics/archive/master.zip** for a detailed overview of the circuit.

Project 1 – Flashing an LED 1001

We start our projects using a simple flashing LED as a means of output. In this era of virtual reality and high-definition gaming, the humble LED is still seen as the "Hello World" of physical computing. The first project with hardware and software should always be simple.



This circuit diagram shows where every component needs to be placed. We used *Fritzing* (**http://fritzing.org/home**) to draw the diagram.



Our project uses one simple circuit built upon a breadboard to demonstrate essential coding concepts and maker skills using Python.

It encourages the learner to attempt it and rewards them quickly with a view to further challenges.

In this first project we shall flash each of the LEDs present on the breadboard. Each LED will flash three times in rapid succession before moving on to the next. Via this project we shall learn to import libraries to enhance our code. We also learn about variables and two types of loops.

To get started, open the Python 3 application from the Programming menu, found in the top-left of the Raspbian desktop. Once this is open click on File > New to create a new blank document. Now click on File > Save and call the file **Project-1-LED-Flash.py**. Subsequent saves will be instant.

Our first two lines of Python will be to import external libraries into our code, namely the **GPIO Zero** and **Time** libraries. Importing libraries is a way of adding extra abilities to your code. For example, by adding the GPIO Zero library we can now interact with the GPIO pins. There are many libraries for Python and we go into detail via the boxout in this tutorial.

From the GPIO Zero library we import **LED**, which this will enable us to interact with the LEDs. From the Time library we import **sleep**, which will enable us to control the pace of our project.

```
from gpiozero import LED
from time import sleep
```

The next section of code contains four variables. Variables are containers for information. They can store anything that we need to store, from a person's name to sensor data. A variable is given a name, typically one that reflects the expected content. In

this project we create four variables:, the first three (red, amber and green) are used to store the number of the pin that will be connected to the long leg of our LED. We use the GPIO Zero LED function to call and store the location, which is then neatly wrapped up in our variable. The fourth variable, **flashes**, stores an integer value, a number that has no decimal place. This integer will control how many times each LED will flash, hence its name.

```
red = LED(17)
amber = LED(27)
green = LED(22)
flashes = 3
```

We now move on to the main loop that will control our project. To run the code indefinitely we use a **while True** loop. This is an infinite loop that will run the code contained therein forever. As you can see in the code, after starting the loop the code underneath is indented. Python uses indents to identify where code belongs. In this case the code belongs in the infinite loop.

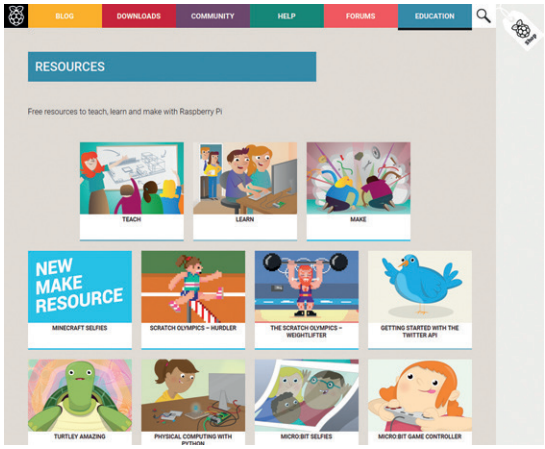
```
while True:
```

We now start a new loop inside the main loop. The new loop, a **for** loop, will use a range to control how many times it goes round. The range is a number that we want to count to. In this case we use the variable "flashes" to control the number of times that we wish to flash the LED, the default value being three times.

```
    for i in range(flashes):
```

The code now indents a further four spaces as we have just created a new **for** loop and now we need to write the code that will be actioned for a certain number of times. This is the code that will flash our LED on and off. Earlier we named each variable after the colour of LED we wish to use. So we need to turn on each colour, wait for half a second using **sleep(0.5)**, then turn off the colour before another sleep is used. This last sleep is important, otherwise we will not see the flash effect. So here we see the code that will flash the red LED, and this is inside of a **for** loop that will iterate three times before it exits.

```
        red.on()
        sleep(0.5)
        red.off()
        sleep(0.5)
```



The Raspberry Pi website has plenty of resources for you to investigate at your leisure. They cover most topics and ability ranges. **www.raspberrypi.org/resources**.

Once the red LED has finished flashing, the **for** loop ends and returns to the next line in the main loop. The next line in this example is another **for** loop that will flash the amber/yellow LED in the exact same fashion as the red LED. This code is indented in the exact same manner as the previous red LED sequence.

```
    for i in range(flashes):
        amber.on()
        sleep(0.5)
        amber.off()
        sleep(0.5)
```

We repeat the same code for our green LED, replacing **amber.on()** with **green.on()**.

With this code complete, save your work and click on Run > Run Module to start the LED flashing sequence. Now watch your LEDs and count how many times they each flash. An extension activity for this project would be to change the number of times the LED will flash, and then change the speed at which it flashes. To stop the code at any time press Ctrl+C

Project 2 – Random choice LED 989

For project 2 we add a random element to our code. In this project we reuse the same circuit as Project 1 but this time we introduce user input in the form of a button. Once the button is pressed it triggers the code to choose an LED at random and then flash the LED.

Click on File > New, to open a new blank document.

Python libraries

In this tutorial we lightly touched upon Python libraries, sometimes referred to as modules. Libraries are a great way of adding extra functionality to your code, but what libraries are included as standard? Well if you would like to find out, open the Python 3 shell by clicking on Run > Python Shell. In the shell type this command followed by Enter.

```
help()
```

You will now see a helpful message giving instructions on how to use the tool. We're interested in the modules, so type modules and press Enter. It may take a few seconds for the list to appear but it will show a complete audit of every Python library installed and ready for use. If you're interested in a particular library then you can learn more by typing its name in the shell. Once you're done with the help tool press Ctrl+D to exit back to the Python shell.

If you need to install a particular library then you'll need to open the Linux Terminal, the icon for which is located in the top-left of the Raspbian desktop. In the new terminal window type

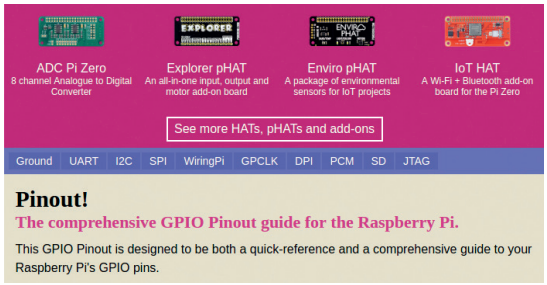
```
$ sudo pip3 search NAME OF THE LIBRARY
```

If you don't know the library name just type a keyword, for example "twitter" to show all the Twitterrelated libraries. Once you have found a library to install, in the terminal type

```
$ sudo pip3 install NAME OF THE LIBRARY
```

This will install the library and further enhance the possibilities of your projects.

If you are a little baffled by the GPIO pins, fear not: **http://pinout.xyz,** Written by Phil Howard from Pimoroni, is your single resource for all of the pins.



Immediately save your work as **Project-2-Random-LED.py** and remember to save regularly.

We start in the same manner as Project 1, by importing a series of libraries to add functionality to our code. But in this project we import **Button** from GPIO Zero. This will enable us to use any free GPIO pins that we wish as a button. We also import the **choice** function from the **Random** library, a function that enables us to introduce a random choice element to the project.

```
from gpiozero import LED, Button
from time import sleep
from random import choice
```

Just as in Project 1 we declare variables to handle the locations of our LEDs, which are attached to specific GPIO pins. But in this project we introduce a button, sometimes called a momentary switch. This button is attached to pin 2 of the GPIO and so using the GPIO Zero library we declare that there is a button attached to said pin. Our last variable, **flashes**, handles how many times the chosen LED will be flashed.

> Printing output to the shell is a great way to understand how the project works, and it helps when debugging

```
red = LED(17)
amber = LED(27)
green = LED(22)
button = Button(2)
flashes = 3
```

Our next line of code requires an introduction. **LEDS** is a list, data that is stored as a list of comma separated values. In other languages a list is sometimes called an "array", but they all work in much the same manner. A list can be used to store lots of values, in this example we use a list to store the variable names used for our LEDS. Lists are indexable, meaning that we can retrieve data from any part of a list, and we can also add and delete data in a list. Our list contains three items of data, numbered 0,1,2, as Python starts counting from zero. So red is 0, and if we wanted to print that value of red we would use **print(LEDS[0])** and it would print the content of the variable **red**. We cover more about Lists in the boxout. So now we create our **LEDS** list and use it to store the LEDs we have attached.

```
LEDS = [red,amber,green]
```

We now enter into an infinite loop, **while True**, which is identical to what we created in Project 1.

```
while True:
```

Our next line of code is a conditional statement, in this case it simply asks whether the button has been pressed, (remember that "button" refers to the variable we created earlier). When the button is pressed the GPIO Zero library understands that the button changes the state of the GPIO pin it is attached to. When left unpressed the GPIO pin connected to the button is pulled high, meaning that it is turned on and flowing with current. When we press the button we connect this high pin to the GND, pulling the GPIO pin low and triggering a change of state which GPIO Zero uses to identify a button press.

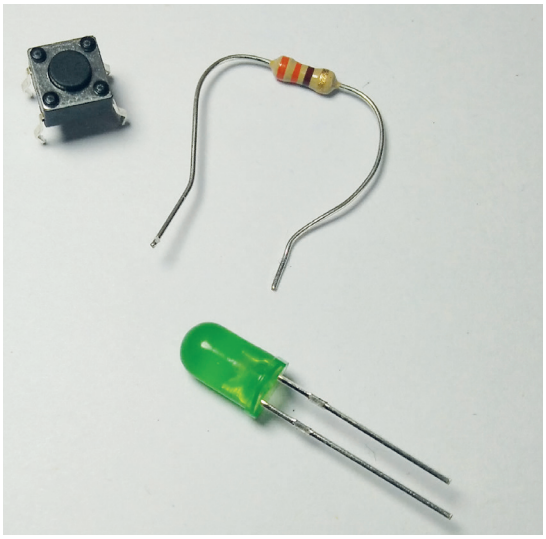
```
    if button.is_pressed:
```

We now need to write the code that will be actioned when the button is pressed; as with Project 1, any code that belongs to a loop or a conditional is indented to identify where it belongs. So when the button is pressed we create a variable called **colour** that will store the answer to a random choice of the LED colours in the **LEDS** list we created earlier. We then print this to the Python shell for debug purposes. Remember that printing output to the shell is a great way to understand how the project works, and it helps when debugging a problem.

```
        colour = choice(LEDS)
        print(colour)
```

Just as in Project 1 we create a **for** loop that will iterate for the number of times stored as an integer in the **flashes** variable. But what's different this time is that we use the **colour** variable that we have just created. The **colour** variable stores the colour of the LED that has been randomly chosen. So rather than explicitly call the LED by its colour we just use the **colour** variable and add **on()** or **off()**.

```
        for i in range(flashes):
            colour.on()
            sleep(0.5)
```



Components such as buttons, resistors and LEDs are cheap and plentiful.

```
        colour.off()
        sleep(0.5)
```

And with this last section of code completed we should now save our work, then click on Run > Run Module to start the code. Now press the button next to the red LED to start the random choice. Due to the small selection of options you will see the same colour chosen quite often – it only has a 1 in 3 chance of being chosen. An extension activity for this project would be to add more LEDs and create a lightshow that runs before the colour is chosen to give the illusion of lots of options being computed.

Project 3 – Conditional LEDs 891

For our final project we shall adapt our code to create a function, a method of running a sequence of code by calling its name. The goal of this project is to press a button and have a corresponding LED light up.

Click on File > New to open a new blank document. Immediately save your work as **Project-3-Pick-A-Colour.py** and remember to save regularly. We start our code by importing the libraries that will provide the GPIO functionality for our LEDs and buttons and control the pace of our project. These are the same as in Project 2.

```
from gpiozero import LED, Button
from time import sleep
```

As with our previous projects we create variables to state the GPIO pin used for our LEDs.

```
red = LED(17)
amber = LED(27)
green = LED(22)
```

Next we create three new variables; these are used for our three buttons, which represent each of the coloured LEDs in our diagram. Each button is attached to a specific GPIO pin which we now declare.

```
red_button = Button(2)
amber_button = Button(3)
green_button = Button(4)
```

The list that we created in Project 2 is re-used to contain the names of the LEDs that we have created as variables.

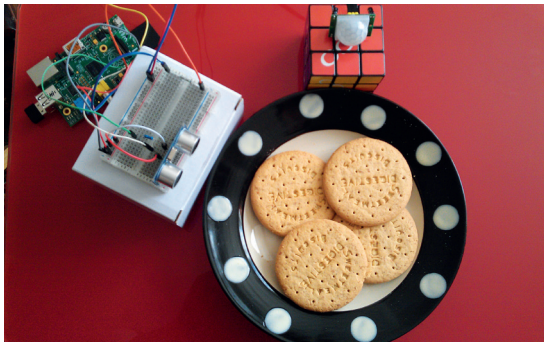
```
LEDS = [red,amber,green]
```

Our next section of code is where we create a function. A function is a sequence of code that we can execute by calling its name in the main body of code. Functions come in two forms, a basic function that when called will execute the code contained within. But the most useful is a function that takes an argument, an extra parameter/instruction that it can act upon. We are going to write a function that will take the colour of the LED that we wish to illuminate as an argument.

We start by defining the name of the function, in this case **lightup**. Inside of the brackets we give a placeholder name for the argument, which is "colour".

```
def lightup(colour):
```

So now our code is indented to signify that it is part of the function. Our first two lines of code in the function are a **for** loop. This loop will ensure that all of our LEDS are turned off before we turn any more on. This loop iterates through all the colours in the LEDS list we created, turning each of them off.



```
    for led in LEDS:
        led.off()
```

We next break out of the **for** loop, but remain inside the function. Our next six lines are a conditional statement that will test to see if the argument passed to the function is one of the three LED colours that we have used. If the colour passed is not red then the test is repeated for amber and green. For each condition we indent the code to show what will happen if the condition is true.

```
    if colour == "red":
        red.on()
    elif colour == "amber":
        amber.on()
    elif colour == "green":
        green.on()
```

The final part of this project is to create an infinite loop that will constantly run a conditional statement to check which button has been pressed. When a button is pressed it triggers the code for that condition to be executed. In this case it will print the colour of the LED to be lit (we do this for debugging purposes). We then call the **lightup** function and pass the argument, being the colour of the LED to be lit.

```
    if red_button.is_pressed:
        print("RED")
        lightup("red")
    elif amber_button.is_pressed:
        print("AMBER")
        lightup("amber")
    elif green_button.is_pressed:
        print("GREEN")
        lightup("green")
```

Our last line of code is outside of the conditional statements but is still inside the main loop. It simply instructs the code to sleep for a tenth of a second between checking the status of the buttons.

```
    sleep(0.1)
```

With the code now complete, ensure that you save your work. Once it's ready, click on Run > Run Module to launch the code and press one of the buttons on the breadboard to trigger the light of your choice.

Les Pounder makes things, breaks things, and spends the rest of his time teaching teachers about the new IT curriculum.

Once you have mastered this project you are ready to move on to our Biscuit Security System project **https://www.linuxvoice.com/raspberry-pi-simple-forms-of-input**.

IMAGE PROCESSING, FIXING AND STORING

Apply smart fixing, processing and storing techniques to your images.

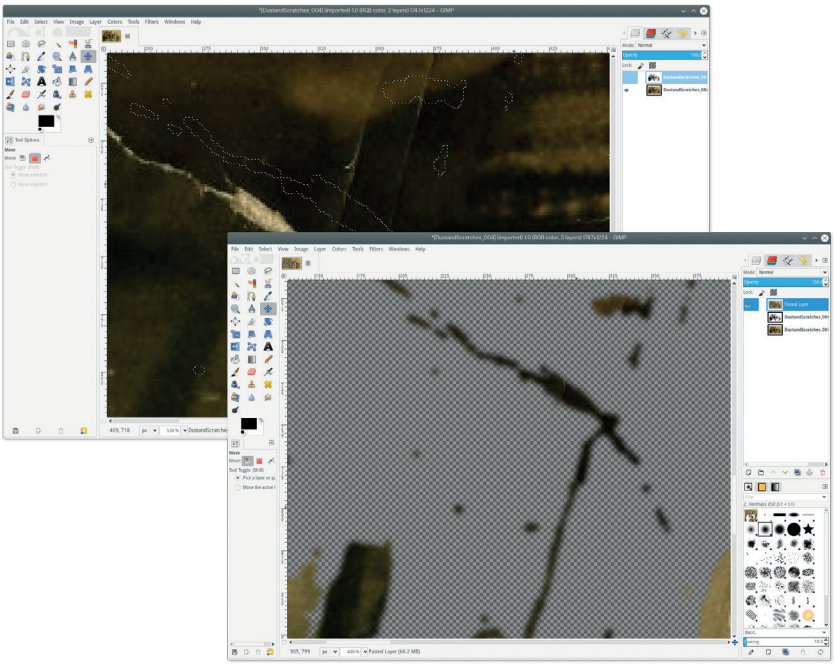
ALEXANDER TOLSTOY

Why do this?

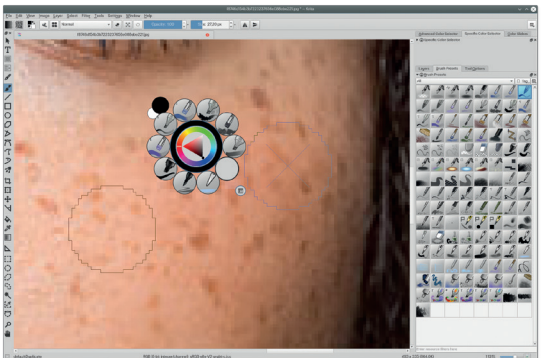
- Perform mundane tasks with batch processing
- Clean up dust and scratches
- Save space with sensible file formats

The days when digital cameras produced terrible results with noise, blur and pale colours – which led us more often than not to *Gimp* to fix things – are gone. Now when most people take point-and-shoot photos with their smartphones, most of the work is performed automatically – cameras apply smart logic to produce perfect JPEGs while various cloud services (starting with Google) upload your gallery online. But there are still other use cases where manual intrusion is needed – scanning and restoring old photographs, playing with creative collages, photo retouching, backing up huge amounts of graphic data and so on. In this tutorial we'll take a look at certain techniques that will help you apply an action to many images at once, fix specific image defects and optimise your storage.

When it comes to image editing, the undisputed king of the hill is *Gimp*. Here we can do almost anything with bitmaps, and we'll start with the clone tool. Click its icon on the toolbar (it's depicted as a stamp), then Ctrl+click the source area (which the tool will copy pixels from), move your mouse to the target area and start painting by clicking and holding the left mouse button. *Gimp* lets you adjust the size, aspect ratio and angle of the brush right in the clone options. Hardness and shape should be adjusted in the brush catalogue, which can be invoked also from within the clone tool options. It's best to press the Edit button and adjust all available brush parameters to match your liking.



Overlap dust and scratches with patches of the 'healthy' neighbouring pixels – that's the essence of the new method.



The healing brush in *Krita* is easy to set up, and it produces great results in retouching skin defects.

Cloning lets you fix certain defects, like dust and scratches, but it is also useful for removing undesired objects from a photo. Say, you've got a marvellous portrait taken on a sea shore, but people on the background deflect the attention of the viewer. Playing with the brush settings for the clone tool in *Gimp* you can gradually remove strangers by accurately cloning sand, water and sky parts and make your photo perfect. By the way, you can do the same in other Linux software too. For instance, *Krita* – a part of the KDE-centric *Calligra* office suite – has the Edit Brush Settings button on its top toolbar, where you can select the clone brush engine and proceed with stamping exactly the same way you did in *Gimp*.

Photographer, heal thyself

Krita and *Gimp* are the most bold and feature-rich Linux graphics applications; now, let's look at their healing capabilities. The healing brush is an advanced version of the standard clone tool. It can be found as a separate tool at the *Gimp* toolbar (it looks like a yellow adhesive bandage), or as an option in *Krita*'s brush engine settings (Painting Mode > Healing). Either application you choose, healing is preformed in the same way as cloning – you set the source area and

transfer it to the target are – but unlike plain cloning, a healing brush respects surrounding pixels and lets you carefully recover texture and colour tone in damaged areas. When working with skin defects healing performs noticeably better than cloning and lets you avoid blurred areas caused by a soft brush.

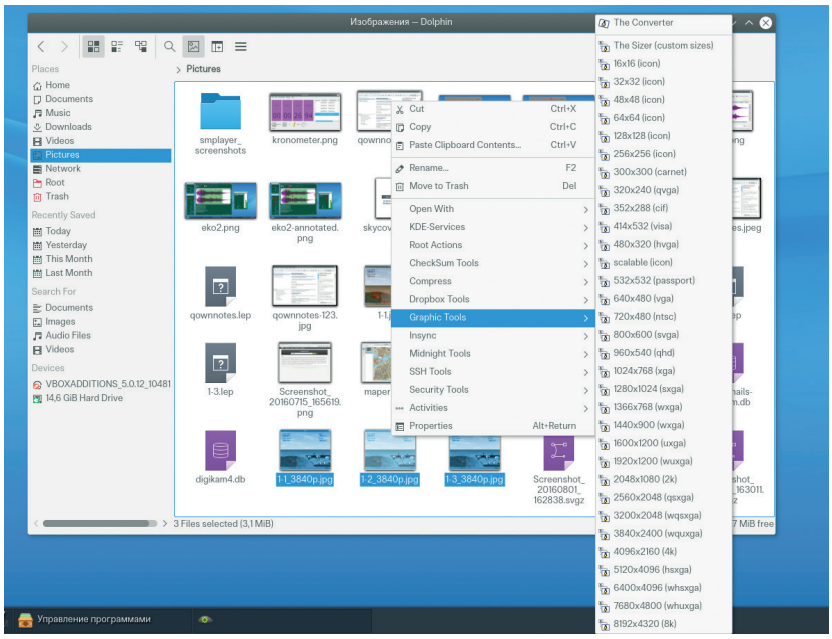
The next big thing we'll do in this section is removing dust and scratches – very common defects that frequently occur in digitised historic pictures. The problem is that you can hardly automate dust removing routines, and in most cases you'll need to diligently remove each spot with dozens (if not hundreds) of cloning or healing actions. Luckily, there exists an original technique that greatly reduces the amount of work. Again, we'll be using the best image editor for Linux – *Gimp*.

Logical programming

The idea comes down to the following steps. First create a copy of the current layer, desaturate it and increase the brightness and contrast to much higher levels, so that dust and scratches will turn to exactly white spots, but midtones will remain at their shades of grey – you'll need to play with brightness and contrast sliders to achieve the best result, as it will be individual for each image. Now choose the Select by Colour tool and click anywhere on a dust or scratch. Hold down Shift and click on a lightest grey pixel somewhere nearby to add more area to your selection. You now have a mask of defects, which includes some undamaged areas as well – don't worry, we'll deal with it later. Now extend your selection by one pixel (Select > Grow) and soften it by some more pixels (Select > Feather). You can now delete the current overburned layer safely – look how your selection matches dust and scratches of the original image Now here's where the magic happens.

Select the Move tool and (important!) change its mode to Selection. It means that when you move anything now, only the selection itself will be moved, but not image content. Move your selection sideway by few dozen pixels, so that the selection now covers mainly 'healthy' areas to the neighbouring damaged spots. Do the copy-paste sequence, change the Move tool mode back to Layer and drag your pasted bits back to their original position. How do you know when you're back in the right place? Well, change the blending mode for the pasted bits from Normal to Difference and let it guide you. When your patches are in place, use the Eraser tool with a reasonably large soft brush to remove false-triggered patches. Finally bring back the Normal blending mode and press the Anchor icon to merge your patches with the background. It all may sound complex, but once you do at least one iteration of such 'patching', you'll love this method because it really saves you a lot of time.

Sometimes you need to apply an edit or transformation to a lot of images, and it makes some users believe they need to do a huge number of routines by hand. Luckily, they need not, and in the following examples we'll find out how to automate bulk actions.



Batch processing is easier than you might think. Don't do so many recurring actions by hand!

The first common action is resizing. Say, you need to attach 20 images to the email, but the recipient doesn't need that many megapixels in each photo, while the web server would simply reject the very big attachment. The easiest way to scale down your photos is to install the dedicated plugin for the *Nautilus* file manager. In Fedora, do it with the following command:

> We'll remove dust and scratches – very common defects that frequently occur in digitised historic pictures

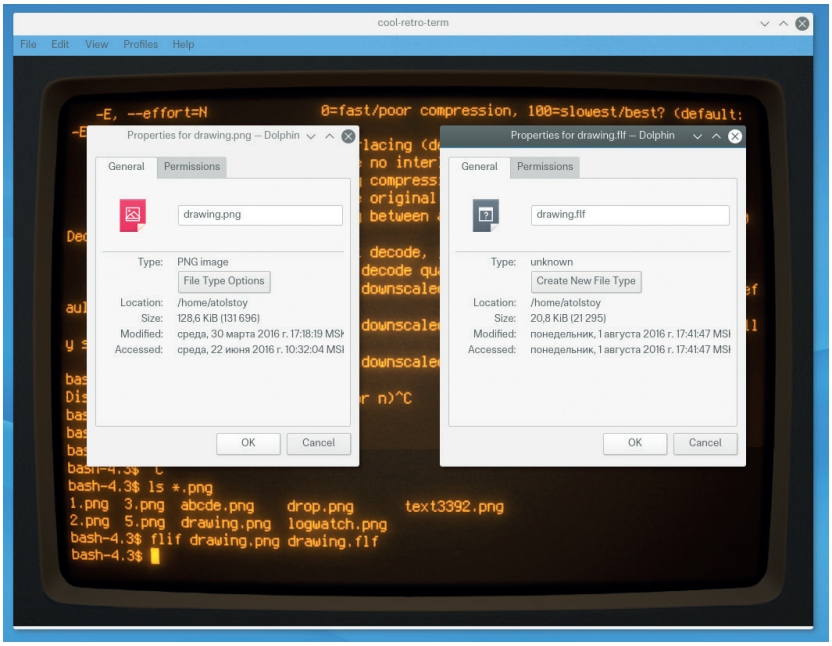
```
$ sudo dnf install nautilus-image-converter
```

There's a very similar (yet different) extension in Ubuntu, which you can get this way:

```
$ sudo apt-get install nautilus-image-manipulator
```

Whichever you'll install, don't forget to restart *Nautilus* (or simply log out and then log back in) in order for the changes to take effect. Once you do that, select the scope of images you'd like to resize, right-click any of them and select Resize Images. Both Ubuntu and Fedora-centric *Nautilus* extensions provide nearly the same features: they let you specify the new size in pixels or percent, choose to save new files along with the originals (non-destructive mode) and also apply a prefix to filenames. As the names suggest, you can also mass-rotate images in the same way – just choose the appropriate item from the *Nautilus* right-click menu.

If you need more features for bulk actions, try *Phatch* — a photo batch processor for Linux. Phatch can batch resize, rotate, apply shadows, perspective,



Squeeze extra disk space by using the latest next-gen file storage formats. Everything is open source!

rounded corners, duplicate directory hierarchies and much more. *Phatch* is available in many Linux distributions, including Ubuntu and Fedora, so install it in a convenient way via **apt-get/dnf** or from within a graphical package manager. Keep in mind that *Phatch* also has a special *Nautilus* integration extension called **nautilus-phatch**. Install it as well in order to access extra bulk actions from images' context menu.

> In real life most photos are stored in JPEG, whereas screenshots and web graphics look best in PNG

In *Phatch*, press the big + sign to add actions to the queue. The list of actions is quite extensive and you can also put several actions in one queue in any combination. *Phatch* lets you be more creative with your images and add, say, a watermark, special EXIF tags and do image transformation with a few mouse clicks. Don't forget to put the Save action at the end of the queue in order to keep the original images safe and secure.

In fact, you don't have to use any desktop environment in order to process images in Linux: In cases when you want to escape from a heavyweight GUI desktop, you can happily deal with images using command line tools. Once you have *ImageMagick* installed on your system, you can manipulate images however you want. Start with the **display** command to view images, like this:

```
$ display path/to/picture.jpg
```

Convert images from one format to another using the following syntax:

```
$ convert sunrise.tif sunrise.jpg
```

Explore the depth of the **$ convert --help** command and find out how to resize and rotate images as well. In the following example we'll convert, resize, rotate our test image and also apply the charcoal filter to it:

```
$ convert sunrise.tif -resize 1024x768 -rotate 180
-charcoal 4 -quality 80 sunrise.jpg
```

If you need to perform bulk actions on the command line, feel free to use *Bash* as a means for applying a command to multiple files. For example, let's rotate all images in the current directory while retaining the original files:

```
$ for file in *.tif; do convert $file -rotate 90 rotated-$file;
done
```

Or maybe we want to convert files to another format and resize them within one command:

```
$ for i in $( ls *.tif); do convert -resize 50% $i $i.jpg; done
```

Use advanced image formats

There are dozens of graphic file formats, but in real life most photos are stored in JPEG, whereas screenshots and web graphics look best in PNG. Let's skip other specialised file formats such as TIFF or numerous RAW flavours from different camera vendors for the sake of brevity. Historically JPEG compression was developed as an optimal trade-off between higher

Sorting and tagging

There are some great open source applications that can help you store and tag images – it's fairly useful once you photo library grows large. The most well-established photo organizers are *Digikam* and *Shotwell*. These are desktop heavyweights and thus are shipped with almost any Linux distributions. Let's have a look on how to sort and tag images in *Digikam*.

When browsing your images in *Digikam*, select at least one of them, right-click and go to Assign Tag > Add New Tag. Enter the name of the new tag and optionally set the keyboard shortcut, which will let you seamlessly assign/clear that tag later on. When you're done, you tag will appear in the Recently Assigned Tags of an image's context menu. Sooner or later you will have many tags that need to be properly managed and reviewed. You can see the full tags tree either through the context menu at Assign Tag > More Tags, or by clicking on the tiny Tags button at the left side panel in *Digikam*. Initially, tags form a plain list, but in fact they are designed to form an hierarchy. If you select a tag and create another tag in its context menu, it will be placed inside that parent tag, so that you can create complex trees, like People > Relatives > Family > Brothers, or, say, Places > Vacation > Summer > America. Of course, it takes a certain amount of time and diligence to add tags manually, but it is worth it. Tags add extra features to your library and simplify image searches: you can browse the convenient list of date-based subdirectories with your images, but once you need to perform custom searches (eg 'show me all pictures of my dad'), tags are your best friends.

Shotwell has a very similar approach to managing tags, and it also has a very useful left-side panel with events, folders and tags tree. Anyhow, *Shotwell* is believed to be a lot simpler than *Digikam*, while being more suitable for novice users. *Shotwell* is more user-friendly and robust, while *Digikam* sports some pretty snazzy features that other photo applications do not, such as automatic face recognition and grouping photos by depicted people.

quality and smaller file size, which used to be a headache in the days of dial-up internet and small hard drives. JPEG is lossy, but it delivers very natural-looking images even with agressive compression ratios. On the other hand, the PNG (Portable Network Graphic) gracefully solved the problem of transferring lossless bitmaps graphics across networks thanks to its built-in compression.

In modern days the technology has advanced even further and now we have two projects on GitHub that outperform the above two formats in terms of compression efficiency: Lepton and FLIF.

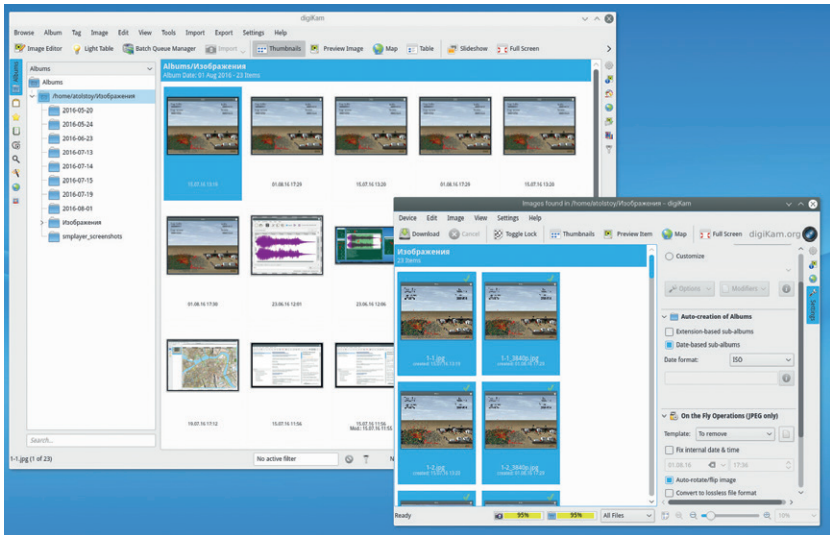
What the flif?

FLIF is a new lossless image format based on MANIAC compression (Meta-Adaptive Near-zero Integer Arithmetic Coding), which outperforms PNG, FFV1, lossless WebP, lossless BPG and lossless JPEG2000 in terms of compression ratio. Let's see how we can use it. Download the code snapshot from the project's website, make sure that you have the **libpng-dev** package (or similar) and then type **$ make**. In a few seconds you'll get the compiled **flif** binary, which acts as an encoder and decoder of FLIF images. The syntax is very simple:

```
$./flif input.png output.flf
```

and the traditional **./flif --help** for the list of options, such as interlacing, palette, quality and more. Keep in mind that FLIF currently can convert only PNG, PAM and PNM file formats, so it makes sense to compare FLIF outputs with the PNG as the most widespread format from the above list. In our tests, lossless FLIF output was smaller than PNG with the highest compression ratio and comparable with JPG in terms of file size. The only problem is that you cannot view **.flf** files, so right now you can only use the **flif** binary to archive images and recover them later using the **$ ./flif -d input.flf output.png** command. However, it is possible to view the FLIF-encoded images using the separate tool called UGUI_FLIF (**https://github.com/FLIF-hub/UGUI_FLIF**), which is based on HTML and JSON. Once the FLIF file format is finalised there will be a more appropriate viewer – because files encoded with an older FLIF binary will not be compatible with a newer one.

Lepton is a compression and decompression algorithm that was open source by the Dropbox company. This cloud-based storage provider has proprietary code in its core, but otherwise it is very friendly to the Linux community. Lepton does with JPEG what FLIF does with PNG, and as long as the two image formats are completely different, we cannot compare their performance directly. Perhaps you might want to use PNG for screenshots and JPEG for photos, and now there are finally two open source projects for that. Lepton offers a lossless compression for JPEGs, which resembles convenient archiving. But unlike putting a JPEG file to a ZIP or TAR.GZ archive, which makes very little sense in saving the disk space, Lepton really makes the output file smaller. Dropbox published a comprehensive article that describes how the algorithm works (**http://bit.ly/29VqPnT**), pointing out optimisations they implemented in encoding 8x8 pixel blocks – these are used in JPEG compression.



Digikam is versatile image sorting machine. Provide it with your images and get the neat subdirectory structure with few mouse clicks.

Store images more efficiently

Building Lepton from source is very straightforward. Download the sources and execute the well-known sequence:

```
$ ./autogen.sh; ./configure; make; make check
$ sudo make install
```

Lepton has a very simple syntax and while you can enjoy many of its advanced features, you can start with this template:

```
$ lepton input.jpg output.lep
```

If your JPEG file uses progressive encoding (eg *Gimp* does it by default when exporting to JPEG), you'll need the appropriate extra option:

```
$ lepton -allowprogressive input.jpg output.lep
```

To restore (decompress) your file, use the same command but switch the **.lep** and **jpg** order. Lepton is quite effective: it can trim down your files by 15–20% without losing quality.

The reason why you might want to use FLIF, Lepton or both is precisely one. Despite the fact that most people have at least 1TB of storage in their desktop machines, the sheer volume of images that have to be stored is growing even faster than most people can reasonably store themselves. Converting some thousands JPEGs to the new Lepton format lets you free extra disk space. Lepton files are about 20% smaller than their source JPEGs, so saving can be be very noticeable. Both Lepton and FLIF are in an early development state, but this only means that we don't have ready-to-use viewers and a desktop integration yet. The core feature — encoding and decoding works flawlessly, so you may want to use Lepton and FLIF as archive tools.

Alexander Tolstoy stores his library of over a hundred thousand cat photos on a single floppy disk.

BUILD A SEARCH ENGINE WITH **APACHE SOLR**

Add some order to the chaos that is the LV catalogue with a Python search engine.

BEN EVERARD

Why do this?

- Build powerful search tools using the technology behind some of the web's biggest sites
- Obsessively tag, index and organise all your things
- Use Python to access RESTful JSON APIs over the internet

When you start *Solr*, you also get a web interface where you can see how everything's running. Point your web browser to http://localhost:8983/solr/#/ to get started.

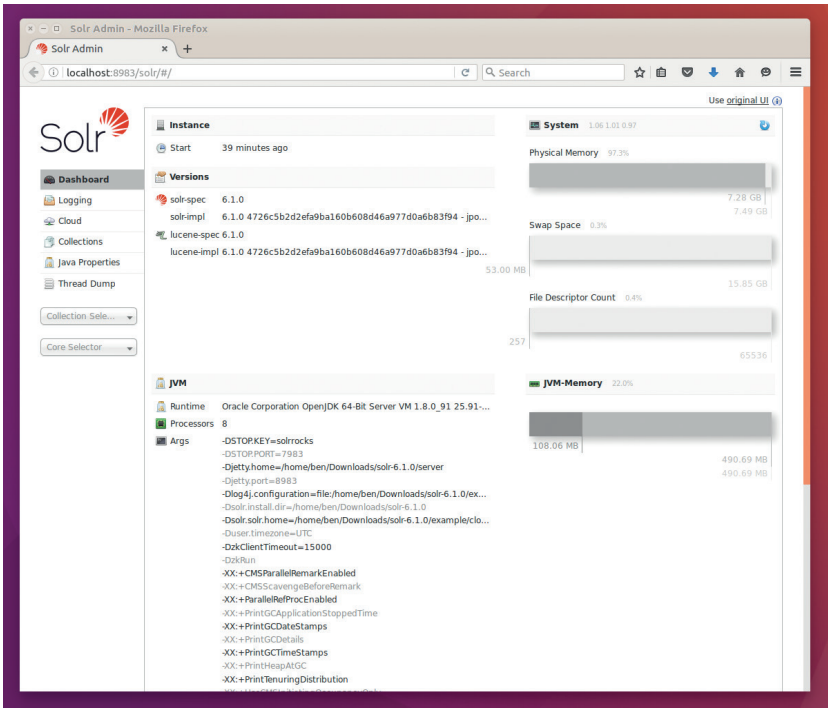
We're producing vast quantities of data. Some of this is nicely structured in spreadsheets and databases, but there's also lots in less computer-friendly formats such as PDF and *MS Word* files. In this code concepts we're going to take a look at using the *Apache Solr* search platform to sift through a trove of PDF files to find exactly what we're looking for.

The first step is to grab the software. The version in most distros' repositories is out of date, so the best option is to download the latest version from **http://lucene.apache.org/solr**. Click on the Download Binary link and you should be presented with a list of mirrors. The file you need should be called **solr-6.1.0.tgz** (or possibly a newer version). As the filename suggests, this is a tarball, so save it to your machine and decompress it.

You can start *Solr* without installing it fully by running the following command in the directory you created by unzipping the download:

```
bin/solr start -e cloud -noprompt
```

This command will start two instances of *Solr* (it's designed to be run as a cluster), one on port 8983 and one on port 7574. We'll be interacting with them via HTTP, so let's start with the most usual HTTP client, the web browser. You can see details of the install by going to: **http://localhost:8983/solr/#/**. We don't need to change anything here for our purposes, but if you have any problems, you can see the logs in the Logging tab.



Solr works with collections. These are groups of data a little like individual databases in a *MySQL* instance. Data is added to a particular collection, and queries are run on collections rather than all the data in the instance. By default, there's a single collection called **gettingstarted**. We'll use this for our project.

Now we have our server and our collection, the only thing we need is data. *Solr* is designed to work with a diversity of datatypes including PDF files, so for our project, we're going to work with Linux Voice PDF files. We release these both for the full issue and for individual articles. We're going to use the individual article files from issue one of Linux Voice, which you can download from **https://www.linuxvoice.com/download-linux-voice-issue-1-with-audio/**. You'll need to download these and save them to a directory that only contains these files. The following command will enter them into the *Solr* data store:

```
bin/post -c gettingstarted <directory containing the files>
```

You can check that everything's worked by querying the search engine via **curl**. The following code should return all the articles that contain the word Linux:

```
curl "http://localhost:8983/solr/gettingstarted/select?wt=json&indent=true&q=linux"
```

This should output a lot of data formatted in a way that makes it hard to read. In order to make our search engine useful, we need to create a human-useable interface for it, and we'll do this with a Python script.

The primary interface for *Solr* is a RESTful API that serves JSON over HTTP. Let's just go back to that previous sentence and pick apart the acronyms before going any further. REST stands for Representational State Transfer, which in itself doesn't tell us much, but it's a style of Application Programming Interface (API) based on the way the world wide web works. Perhaps the most important thing to understand about RESTful APIs is that the server is stateless – that means that all the details

needed for the request are in the request itself (including any login or session information).

The request is served over HTTP, which is the same protocol that web browsers use to get data from web servers, so we can use exactly the same tools. In the first example, we used *Curl*, but we can also use a web browser. Most programming languages have a library for getting things via HTTP (in Python it's **urllib**), and you should be able to use any of them if Python isn't your thing.

JavaScript Object Notation (JSON) is a way of structuring data in key-value pairs in plain text that looks remarkably similar to Python's dictionary format – so similar in fact that we can use a module called **simplejson** to convert the text returned by the RESTful API directly into a Python dictionary.

Using this RESTful API, we can build a really simple query interface with Python:

```
import urllib2
import urllib
import simplejson

base_url ="http://localhost:8983/solr/gettingstarted/
select?wt=json&q="

while True:
        query = raw_input("Query:")
        result = urllib2.urlopen(base_url+urllib.
quote(query))
        result_openned = rsp = simplejson.load(result)
        for doc in result_openned["response"]["docs"]:
                print "file" + doc["resourcename"][0]
+ " pages " + str(doc["xmptpg_npages"])
```

This builds a HTTP request using input from the user, then formats the returned JSON data nicely for printing out.

So far, we've used the query to search through the article text and find things related to our search term, but we can do more than this. *Solr* is a general search engine that works with text documents (like the PDFs we're using here), but it can handle a wide range of different bits of data. Each different type of data contains different attributes. For example, in the Python code above you can see that we print out two items that are returned in JSON: the **resourcename** and the **xmptpg_npages** – the name of the file and the number of pages. Rather than just searching through the text of the article, we can focus our search on one particular attribute. For example, if you enter the following query when running the Python program above, you will get only the articles with two pages:

```
Query: xmptpg_npages:2
```

You can chain parts of your queries together by separating different search terms with spaces. By default, *Solr* will return results that match any one of the search terms. So the following query will match anything that is about either LUGs or FOSDEM:

```
Query: lugs fosdem
```

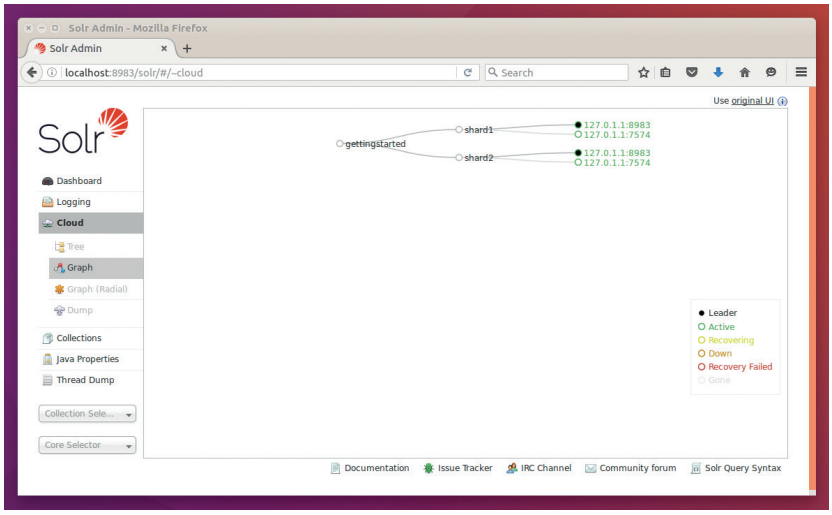
Adding a plus sign to the start of a search term tells *Solr* that you only want results that include that particular term. If you want only those articles that are two pages long and about FOSDEM, you can use the following query:

```
Query: +fosdem +xmptpg_npages:2
```

By default, *Solr* will see how well each document matches the search query and return the results based on their relevance. We can find out just how good a match each document is by getting *Solr* to return the value for score with the **fl** (field list) parameter to our request. We want all the fields that exist (which we get with the ***** wildcard), and the **score** pseudofield. In your Python program, change the **base_url** line to the following:

```
base_url ="http://localhost:8983/solr/gettingstarted/
select?wt=json&fl=*,score&q="
```

You can now include the score in the final print line of the script:



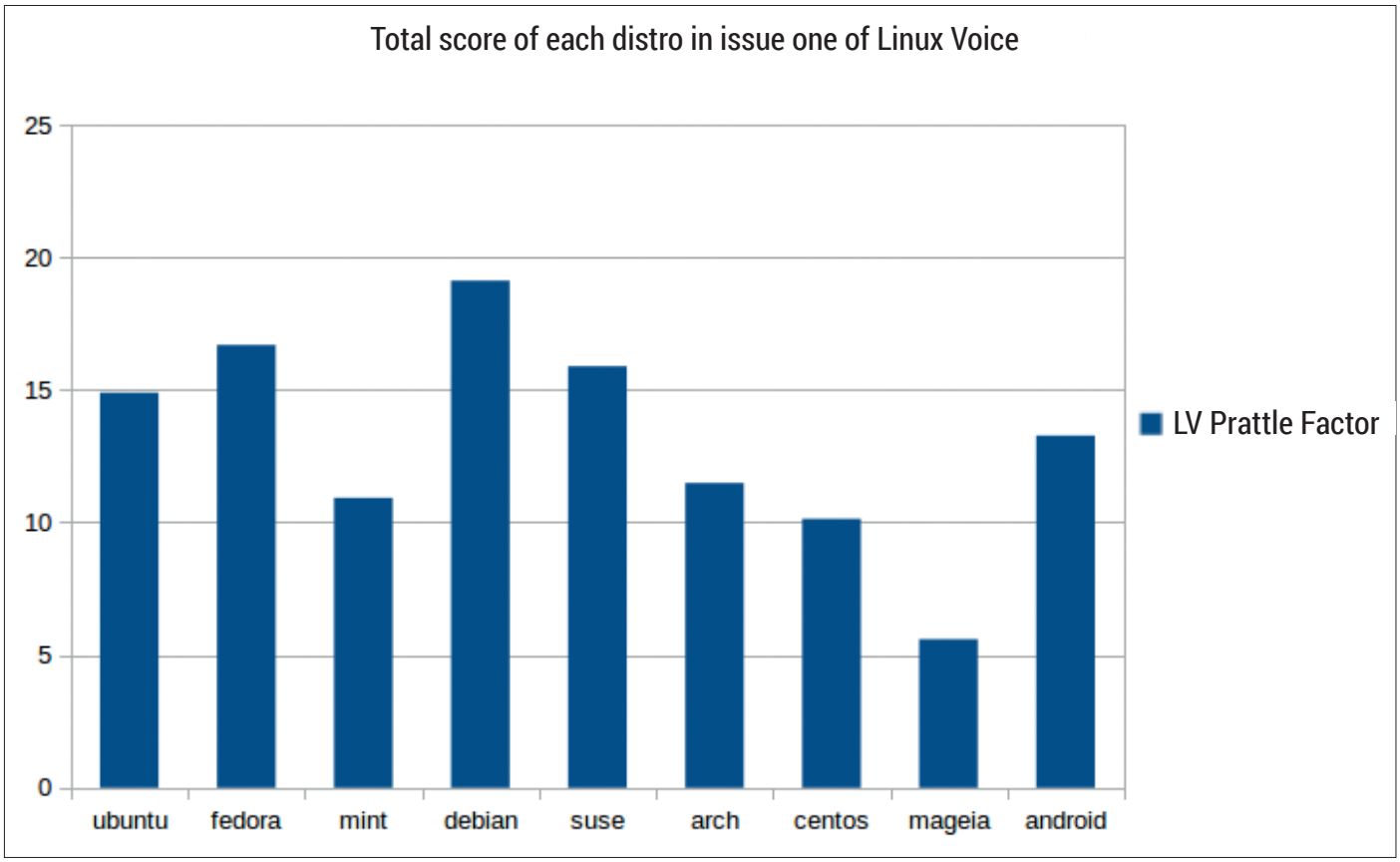
Solr is designed to be scalable to huge instances so it runs with shards by default. In the basic setup, two shards run on the same machine, but in production there could be multiple shards across many data centres.

> Solr is a search engine that works with text documents, but it can handle a wide range of different bits of data

Apache Lucene and Solr

If you're just after a text search library to add to your project, you can use *Apache Lucene*, which powers the search in *Solr*. Although it's written in Java, there are *Lucene* bindings for other languages including Python that enable you to bring in just the searching power to your project. The advantage of *Solr* is that you get the full server – with the ability to run in a distributed manner – to store your data and handle queries.

Solr itself started as an internal project by CNET Networks, which donated the code to the Apache Software Foundation in 2006. Under *Apache* the software has developed significantly particularly in its ability to handle huge numbers of queries very efficiently. It's proved itself on some of the internet's biggest sites including the Internet Archive (**archive.org**), **DuckDuckGo** and Netflix.



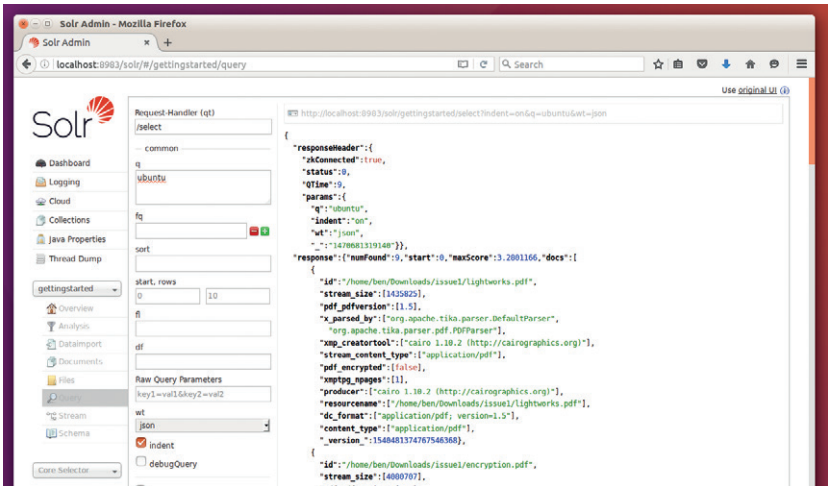
Fedora was the most relevant distro to issue 1 of Linux Voice.

```
print "file: " + doc["resourcename"][0] + " pages " +
str(doc["xmptpg_npages"][0]) + " relevance " +
str(doc["score"])
```

This is a useful way to find out just how good a match a particular page is, and it can also be useful in determining how predominant a particular word or phrase is in your collection. If we add up the score from every document returned by *Solr*, this will give us an idea just how dominant that word was in issue one of Linux Voice. We'll call this the LV Prattle Factor because it says how much we prattle on about it. The code to add this feature is:

```
import urllib2
import urllib
import simplejson
base_url ="http://localhost:8983/solr/gettingstarted/
select?wt=json&fl=*,score&q="
while True:
        lv_prattle_factor = 0
        query = raw_input("Query:")
        print base_url+urllib.quote(query)
        result = urllib2.urlopen(base_url+urllib.
quote(query))
        result_openned = rsp = simplejson.load(result)
        for doc in result_openned["response"]["docs"]:
                print "file: " + doc["resourcename"][0]
+ " pages " + str(doc["xmptpg_npages"][0]) + " relavence "
+ str(doc["score"])
                lv_prattle_factor += doc["score"]
        print "Linux Voice Prattle Factor: " + str(lv_
prattle_factor)
```

The *Solr* web interface includes a query builder to help discover the various options to the HTTP API.



We can now use this to work out one of the age-old questions: just which distros get the most coverage in Linux Voice? You can see the results of this in the graph above – and Fedora was the most talked about distro for us back in February 2014.

Hopefully, you've now got an idea of how to start working with *Solr*, but it can do far more than we've had a chance to look at here. As well as searching, you can look at aggregate data, and perform more complex inquiries using more specialised data, such as location tagging. If you need a customisable search engine, *Solr* is a great place to start.

Ben Everard, Linux Voice's editorial overlord, is probably breeding an out-of-control slug plague as you read this.

MEDECINS SANS FRONTIERES
DOCTORS WITHOUT BORDERS

The world's A&E Department

Find out more at msf.org.uk

FLATPAK: BUILD A CROSS-DISTRO PACKAGE

Why rely on a distro's package maintainers when you can package software yourself?

VALENTINE SINITSYN

Why do this?

- Understand the twisted logic behind dependency hell
- Make it easier for users to get hold of your software

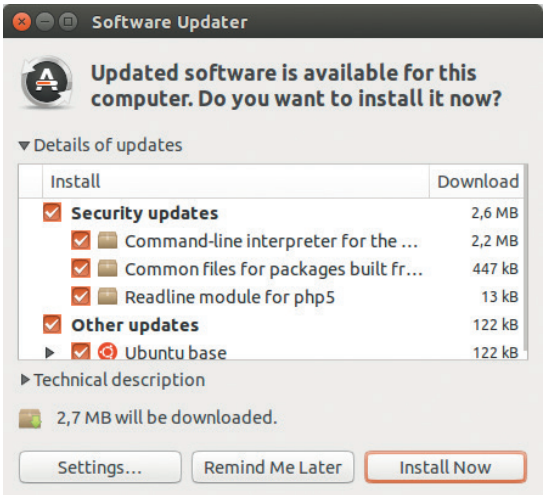
Linux is diverse, and so are its packaging technologies. We aren't going to cover all of Deb, RPM, Pacman, Flatpak, Snap etc in this tutorial. It wasn't planned as a definitive guide to these technologies either. Instead, we'll focus on typical approaches and concepts, using two popular implementations. A package is essentially an archive (often a tarball) containing binary and data files which constitute an application, along with some metadata.

Metadata carries essential information about a package. For instance, it tells us the package name, author, and version. Using these, package management tools (such as **rpm**, **dpkg**/**apt** or **pacman**) can tell if there is a newer version available in the distribution repositories. If you're downgrading a package, the system also understands it and acts accordingly. Metadata can also tell us to which category the package belongs, much like a "Shelve into" hint on a book back cover, and what the package priority is. But most importantly, metadata conveys dependency information.

The software you write (or use) doesn't exist in a void. If it's compiled, it often relies on libraries. If not, it obviously wants an interpreter. For example, you'll need a web server, a database and PHP 5.4 or better to run *OwnCloud*. *Exiv2* and an LDAP server aren't strictly necessary, yet they add some features (namely, Exif and centralised authorisation support).

Metadata encodes all this information in a machine-readable way. The *OwnCloud* package will depend on the PHP package (exact names differ per distribution) with the version not less than 5.4. Both *Apache* and *Nginx* can run *OwnCloud*, so the web server dependency may come through a metapackage. This is a virtual package (think alias) that many real ones may choose to provide. Both *Exiv2* and the LDAP server may come as optional (or suggested) dependencies. When you instruct your package manager to install *OwnCloud*, it pulls in PHP, a database, and whatever web server you choose automatically. This is not the case with optional packages; the tool would normally list them for you after installing *OwnCloud*, and if you'd like any of them, issue a separate command.

It is fully possible to download a package file and install it manually. It implies you'll also download and install all the dependencies, of course. This is tedious and error-prone, so most distributions and even third-party vendors such as Atlassian or Dropbox build package repositories, which are generally reachable over the internet.



From time to time, package managers grab remote repositories listings, and check if there's anything new.

Official repositories are usually self-contained; that is, they provide all dependencies for any package hosted. From time to time, your Linux system grabs the list of packages from every repository you have configured and suggests that you install updated ones. You may also want to refresh package lists manually, for instance, if you configured a new repository. This is what happens when you type **apt-get update** on a Debian-based system.

Debian way

Perhaps that's enough theory for now, and it's time to get your feet wet. Though I'm reluctant to offend the Red Hat users, I'd still say that Debian's packaging format (Deb) is predominant in Linux landscape at the moment. So, let's spend a minute learning how you package your code for Debian and its derivatives.

Before we proceed with our experiments, we need an application to package. Recently I stumbled upon *QOwnNotes* (**www.qownnotes.org**). If you ever tried to use *Evernote* in Linux, you'd probably agree that the experience could be better. *Evernote* forces you

to store documents in the cloud, with all the usual concerns about privacy. And it has no native Linux client. *QOwnNotes* may have fewer features, but it's free (as in speech), it runs on Linux, and it plays well with *OwnCloud*. If something doesn't work for you, you can just go and fix it rather then grumble on the *Evernote* forums. In other words, it's not a drop-in replacement for *Evernote*, but it looks promising.

Moreover, *QOwnNotes* is already packaged for many Linux flavours, including Debian. We'll study this Debianisation and discuss alternative ways where appropriate. I also suggest you look in **QOwnNotes/build-systems** at its GiHub page (**https://github.com/pbek/QOwnNotes**) to check how the same concepts are applied across different packaging tools.

Grab the latest sources as described on *QOwnNotes*' installation page (**http://www.qownnotes.org/installation**). Debianisation lives under **debian/**, which contains a handful of files.

Let's start with **debian/control**. It can be rather long and describe multiple packages, but for *QOwnNotes*, it spans only 18 lines. A blank line splits this file into two paragraphs referring to the source package and the only binary package it creates.

```
Source: qownnotes
Priority: optional
Maintainer: Patrizio Bekerle <patrizio@bekerle.com>
Section: editors
Build-Depends: debhelper (>= 9), cdbs, qt5-qmake,
qtbase5-dev, libqt5svg5-dev, qtdeclarative5-dev
```

Source and **Maintainer** define the package and the maintainer names. Both are mandatory. You also see **qownnotes** is an optional package that belongs to the **editors** category. Other priority levels include **required** (absolutely necessary) or **important** (expected on any self-respecting Unix system).

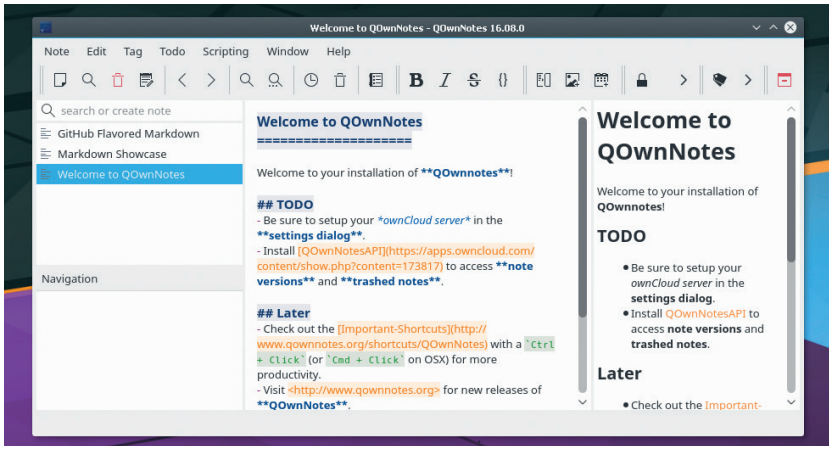
Build-Depends lists packages that are required to build *QOwnNotes*. They aren't necessary in the the same you need to run it. Remember that Debian puts header files and alike into **-dev** packages. Neither **qmake** nor the *Qt* headers are essential for *QOwnNotes* to run, but they are required for the build process. If you wonder what on Earth **debhelper** is, we'll get on it shortly.

A Debian source package is just a tarball (or two) along with a **.dsc** file storing the metadata. The exact format is usually set via **debian/source/format**. *QOwnNotes* uses **3.0 (quilt)**. It was designed for official packages which separate upstream sources and local Debian changes. For us, it's not the best choice, so set the format to **3.0 (native)** now.

A binary package paragraph looks very similar:

```
Package: qownnotes
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends},
libqt5core5a, libqt5gui5, ...
...
```

Package is the binary package name. **Architecture** tells us what hardware architecture the package supports (**any** means it should compile on any host). In rare cases when the program is hardware-specific, you should provide an architecture string like **amd64** or **arm** here. Architecture-independent packages such as your Python code can use **all** instead.



Not an *Evernote*, or a clone, but it still looks interesting to try. Or, at minimum, to package.

Depends lists runtime dependencies. Note there are no **-dev** packages here anymore. Version requirements come in parenthesis: **foo (>=1.0)** pulls **foo** 1.0 or higher. Optional dependencies may use **Suggests** or **Recommends**. Add **Provides** if you provide some virtual package (aka metapackage).

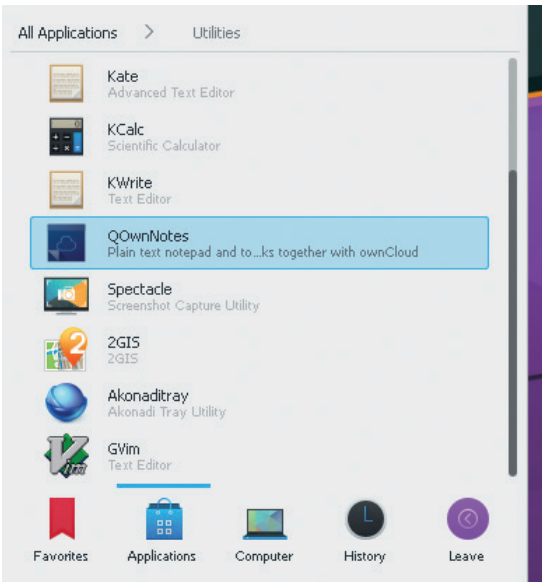
${} is the **control** file syntax for substitutions. **shlib:Depends** runs the **dh_shlibdeps** tool to calculate shared libraries' dependencies of your application. You'd want to include it unless you build a static executable. In a similar vein, **misc:Depends** accounts for the additional packages that **debhelper** may have decided to pull behind the curtains.

debian/control is perhaps the most important piece of package metadata, but not the only one. Did you notice it doesn't tell the package version? This comes through **debian/changelog**. This file follows a strict format described in the Debian Policy Manual 4.4 (**https://www.debian.org/doc/debian-policy/ch-source.html#s-dpkgchangelog**). Usually, you update the changelog with **dch** tool to avoid pitfalls. Last but not least, each package must contain a copyright notice – this is stored in **debian/copyright**, which is both machine and human-readable, like Markdown.

> A Debian source package is just a tarball or two along with a .dsc file storing the metadata

Building binaries

Now, your package wants a recipe to build the code. The main build script for a Debian package resides in **debian/rules**. In a nutshell, it's just a Makefile, which is the *de-facto* standard to convey build instructions in Linux. Here, it's executable and carries a **#!/usr/bin/make -f** shebang. Packaging tools expect you provide some predefined targets in this file, such as **build** to



QOwnNotes comes through Flatpak; everything else is from native packages. Can you tell the difference?

prepare and compile the sources, or **binary** to create a binary package. The Debian Policy Manual, 4.9 has a complete list.

Thanks to *Autotools*, *CMake* and friends, most Linux applications build in standard ways. This means nearly identical **debian/rules**, each taking time to author and debug. Reinventing the wheel is hardly an option, so Debian introduces **debhelper** tool to automate standard bits. **Debhelper** comes with many **dh_*** commands to wrap up the process. You just need to call them from your **debian/rules**.

In the case of *QOwnNotes*, this file is even simpler:

```
#!/usr/bin/make -f
#export DH_VERBOSE = 1
DPKG_EXPORT_BUILDFLAGS = 1
include /usr/share/dpkg/default.mk
export QT_SELECT=qt5
include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/makefile.mk
include /usr/share/cdbs/1/class/qmake.mk
```

Basically, this just includes the CDBS (Common Debian Build System) components needed to build a *QMake*-based application. CDBS is a Makefile "library" which complements **debhelper**. *Qt 5* is selected if there is more than one version of the library installed. **DPKG_EXPORT_BUILDFLAGS** tells us to export **CFLAGS**, **CPPFLAGS** and alike. Uncomment the **DH_VERBOSE** line to make **debhelper** more chatty.

From time to time, backwards-incompatible changes are made to **debhelper** (and most other software as well). So packages should tell which **debhelper** "compatibility level" they expect. It comes through **debian/compat**, and packages set it to 9.

Some parts of your package may not come as a build process artefact. Imagine you want some custom config file or an icon: you can use **debian/install**, or its per-package equivalent, **debian/<package name>.install**, to put arbitrary files into the package. Similar mechanisms exist to create empty directories, install init or cron scripts, and so on.

Building the package is just a matter of typing **dpkg-buildpackage -us -uc**. The switches ask not to sign anything with *GnuPG*. Recall that the code compiles against the libraries (and architecture) you have on your build system. That's how the package is built for the particular distribution, and that's why it's important to have a clean reference environment. Tools like *Pbuilder* or *sbuild* do so via chroots.

Baking Debs is quite easy, but creating proper Debian packages could be tricky. Be sure to read the Debian New Maintainers' Guide (**www.debian.org/doc/manuals/maint-guide**), if you're seriously into it.

All in one

You might call traditional packaging dependency-centric. The idea is to have all the required bits only once per system. This sounds reasonable: why do you want multiple copies of the same library consuming disk space? What if the library appears vulnerable? Keeping track of these multiple copies and updating them is tedious, at best.

The coin is double-sided, though. In the real world, libraries have bugs, and different minor versions aren't necessarily interchangeable. Imagine you have AppA, which wants libfoo=1.0.2, and AppB, which pulls libfoo=1.0.1. This is a famous dependency hell, a rare situation which is nevertheless hard to resolve.

Storage is cheap these days, and isolation mechanisms in recent Linux kernels limit potential damages in case of vulnerability exploitation (but see the boxout). This gave rise to technologies that build self-contained, distribution-neutral packages.

Flatpak (**http://flatpak.org**), formerly xdg-app, is a good example. Flatpak aims at desktop applications and uses much the same sandboxing mechanisms as in container virtualisation.

The self-contained application model could be interesting to proprietary software vendors, but also to complex open source projects such as *LibreOffice* (**https://www.libreoffice.org/download/flatpak**). We've introduced Flatpak back in LV030, and this time, we'll walk through the package building process.

Pack with Flatpak

Before we proceed, make sure you have Flatpak installed. (Traditional) packages are already available for some distributions. We'll stick to Arch, which has Flatpak in the official repositories. If you use Fedora or Debian, the differences would be minimal.

Flatpak runs applications on top of shared runtimes and uses SDKs to build packages. There is an official Gnome runtime, but *QOwnNotes* is a *Qt* application. The experimental *Qt 5*/KDE Frameworks 5 runtime and SDKs are available at **https://community.kde.org/Flatpak**. Follow the instructions on the page to install them: it's easy. Remember, Flatpak sandboxes everything, so it doesn't matter if you have KDE or not.

There are two approaches to Flatpak packaging. Large real-world applications use **flatpak-build** and

JSON manifests to automate the process. *QOwnNotes* is not that big, so we'll do everything manually with **flatpak build**. First, create a directory where you'll build the package. Then, untar *QOwnNotes*' sources and initialise the build process with **flatpak build-init**:

```
$ flatpak build-init qownnotes org.qownnotes.QOwnNotes org.kde.Sdk org.kde.Platform
```

This will create the **qownnotes** staging directory for package files. Each Flatpak application must have a unique name consisting of three or more dot-separated alphanumerics. Otherwise, it could be anything, yet "reversed DNS" notation is most common. Then we specify an SDK and the runtime the application will use. Confusing their order leads to cryptic error messages, so be careful.

Now you can start a build process. Issue commands you'll normally need to build the application, but prefix them with **flatpak build <path to the staging directory>**:

```
$ cd qownnotes-X.Y.Z
$ flatpak build ../qownnotes qmake PREFIX=/app QOwnNotes.pro
$ flatpak build ../qownnotes make
$ flatpak build ../qownnotes make install
```

QOwnNotes employs a *QMake*-based process. Note we instruct it to install the application under **/app** inside the sandbox. This is a requirement.

Some changes are due before you can wrap up the build. Certain files, such as desktop entries or icons, are exported to the host filesystem for integration. Flatpak expects them to begin with an application name to ensure uniqueness. **flatpak-builder** handles the renaming for you, but we're on our own:

```
$ find ../qownnotes/files/share/{applications,icons} -type f | perl-rename 's/QOwnNotes\./org.qownnotes.QOwnNotes./'
```

perl-rename may come as **rename**, if you are not on Arch. You'd also want to adjust the icon name inside the desktop file. Then you'll need to decide which resources the sandbox will share with the host system:

```
$ flatpak build-finish ../qownnotes --socket=x11 --share=ipc --share=network --filesystem=home --command=QOwnNotes
```

QOwnNotes draws windows over the X11 protocol, thus the first two switches. Besides, it will need network access for *OwnCloud* integration, and you'll want to store notes in your home directory as well. The final switch tells Flatpak which command it should execute in the sandbox. Flatpak stores packages in OSTree repositories, which are in turn modelled after *Git*. The next command creates one under **repo** and commits the application:

```
$ cd .. && flatpak build-export repo qownnotes
```

If you receive complaints about desktop file contents, just do the changes as requested. It's possible to wrap the repository into a single-file bundle (see **flatpak build-bundle**). We don't want to do this, as we can get *QOwnNotes* directly from the repository:

```
$ flatpak --user remote-add --no-gpg-verify --if-not-exists qownnotes repo
$ flatpak --user install qownnotes org.qownnotes.QOwnNotes
```

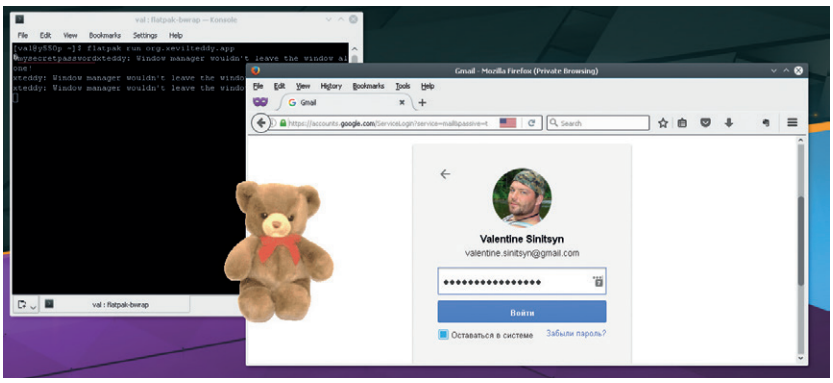
We configure `repo` "remote" repository under the **qownnotes** name and install the application from it. **--user** tells Flatpak this should be done for the current user, not system-wide.

That's it! You can now run *QOwnNotes* with **flatpak run org.qownnotes.QOwnNotes**. It should also be available in your desktop application menu.

Universal packages may help vendors to reach wider audiences. Opponents argue that they make pushing adware or other malware to users easier. Moreover, Linux distributions are diverse for a reason. They target different users, and maintainers oversee this process. Maintainers help to keep software in distributions consistent. And they mediate between users and developers. This is a social role that a technology is hardly able to take.

Perhaps the best thing universal and traditional packaging can do is to complement, not to substitute each other. And now you have some grounds to put both in your arsenal.

Valentine Sinitsyn is a university professor, Mandriva contributor, physicist and Linux systems developer.



It's sooo cuuute… and steals your passwords (red line) even if running in a sandbox. Thank you, X11.

In security, there is no silver bullet

Shortly after Ubuntu 16.04 LTS, featuring the Snap package manager, was released, a proof of concept exploit was published: a cute teddy bear silently grabs your keystrokes in an X11 environment. If it were real, an attacker could use it to steal your passwords or credit card numbers, not to mention personal messages.

With a few tweaks, you can run the same exploit under Flatpak (see the screenshot). It doesn't play well with *Chromium*-based browsers though – you always patch Linux viruses to make them work for you, you know. But it's just a proof of concept, after all.

The problem isn't in Snap (or Flatpak). The vulnerability is in X11 itself, which officially allows any application to receive keystrokes or inject events into any other application. So, the exploit would work in a traditional package as well, yet such package would hardly get past the "maintainer barrier". The bottom line: nothing protects you, if you don't protect yourself.

Valentine Sinitsyn develops high-loaded services and teaches students completely unrelated subjects. He also has a KDE developer account that he's never really used.

CORE TECHNOLOGY

Prise the back off Linux and find out what really makes it tick.

Network Addresses Assignment

Virtually all PCs (and some toasters) seem to have an IP address today. Computing devices aren't born with an IP address (neither v6 nor v4). Before they connect to the outside world, they need this address configured somehow. In the simplest case, you assign it manually. This isn't very convenient from the end-user perspective, however. When we connect to a Wi-Fi hotspot, we expect to start browsing and chatting, not changing settings. Assigning an address can be a convoluted process, and this is what we'll speak about today.

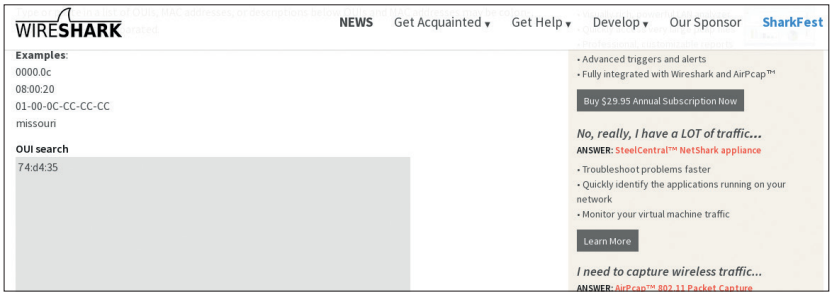
Prehistoric RARP

Now, have a look at your home router's back cover, flip your laptop or take the lid off your smartphone. Chances are, you'll see a sticker with the device's MAC (Media Access Control) address. Usually, it looks like six colon-separated octets: `70:5a:b6:59:c4:5d`. Both Ethernet and 802.11 (Wi-Fi) devices have it. The first three octets tell you the manufacturer; for example, `00:05:5d` is D-Link. These octets are often called the OUI, or Organizationally Unique Identifier. The manufacturer further assigns the remaining three octets, so that each device on the planet gets its own unique MAC. Even virtual network adapters have their own MAC addresses:

```
$ ip -o link show dev docker0
6: docker0: ... link/ether 02:42:11:c8:7c:c4 ...
```

Here the `02:` prefix designates a locally administered address, which doesn't constitute an OUI.

There are many services that let you look up the manufacturer for OUI straight in your browser. Being a Free Software proponent, I recommend the *Wireshark OUI Lookup Tool* found at **https://www.wireshark.org/tools/oui-lookup.html**.



With *Wireshark*'s OUI Lookup Tool, you can check the manufacturer of any network card straight from the browser.

MAC addresses aren't what you use on the internet, though. They naturally depend on the underlying technology, and would be a pain to use in the diverse world that is the internet. Another abstraction layer (IP) is needed. MAC addresses are good to deliver data to their ultimate destination in local networks. This suggests there should be some means to convert between IP and MAC addresses.

Address Resolution Protocol, or ARP (RFC 826) serves these purposes. Hosts learn IP addresses of their neighbours via broadcasting. An everyday equivalent of this would be a delivery boy shouting "Who is Joe Doe here?" in a crowded open space room. This procedure happens regularly on any network, as **tcpdump** may show you:

```
$ sudo tcpdump -n arp
...
14:32:23.852185 ARP, Request who-has 192.168.101.149 tell
192.168.101.44, length 28
14:32:23.853860 ARP, Reply 192.168.101.149 is-at
74:d4:35:56:7d:ed, length 46
...
```

Remember not to run network sniffers on corporate or public networks – this could be deemed illegal.

When a reply is received, the requesting host remembers it in a cache. This helps to avoid broadcasts for known addresses, thus saving you some bandwidth.

What's in that cache on your local system? The **arp** command knows, and it can also manage the cache (which is rarely needed). By default, **arp** prints hostnames. If you want the raw IP addresses, add **-n** to the command line:

```
$ arp -n
Address      HWtype HWaddress       Flags Mask  Iface
192.168.101.44 ether  90:4c:e5:c7:89:69 C       enp1s0
192.168.101.1  ether  04:8d:38:c2:88:b3 C       enp1s0
```

Any user can list ARP cache entries, as they are no secret. You'll need root permissions to set (**-s**) or delete (**-d**) them, however. Try it. You'll notice that some entries, such as the one for your default

gateway, aren't easy to remove. This is because your PC communicates with these hosts quite often. So, it learns the address back before you type **arp**, making an entry re-appear.

ARP maps IP addresses (which are configurable) to MAC addresses (which are built-in). What if we look at this from another angle? Can't you build an inverse of this mapping to obtain IP addresses by the given MAC? Reverse ARP (RFC 903) was historically first to implement this idea. If you think about it for a moment, you'd agree this protocol already handles automatic IP address assignment. An administrator may configure reverse ARP mapping on the router, and network hosts will be able to learn their IP addresses via RARP requests. This works, and early Sun workstations happily followed this approach. Yet it wasn't very flexible. In plain RARP, you can't learn anything but your own IP address: no hostname, no routes, no other settings. Moreover, it's IPv4-only and low-level. For a diskless workstation, knowing the IP address alone won't be sufficient to boot it: you'd also want a path to the boot image. A more advanced protocol was due to solve these issues.

Medieval BOOTP

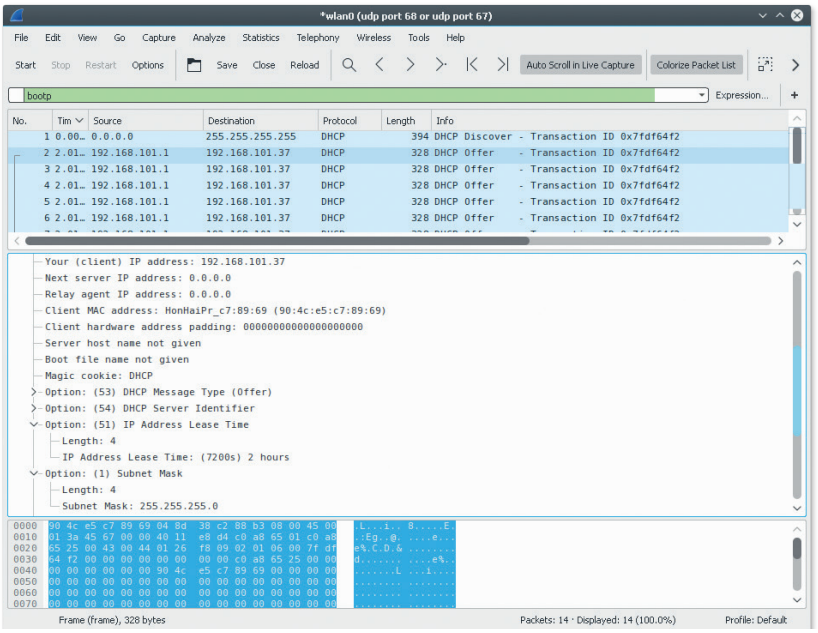
One of the first attempts to build such a protocol was BOOTP (RFC 951). BOOTP is a simple client–server protocol that builds on UDP as a transport. Clients use UDP port 68, whereas servers listens on port 67.

As the name suggests, BOOTP was conceived to facilitate bootstrapping. It defines two message types (or opcodes): BOOTREQUEST for client requests and BOOTREPLY for server responses. Both are using the same package format. In the request, the client says what its hardware address is. In the reply, the server tells the client what IP address it should use. The server also specifies the boot image that the client can download via TFTP. Moreover, the packet structure allows for optional extension fields, which may carry additional bits of data.

The client can also build a request the way it will be handled by the specific server. BOOTP messages carry the server's name, and if the server accepts a packet with a name different from its own, it silently discards it. This adds some flexibility, but it is another bit of data that must be pre-configured on the client.

In a similar fashion, BOOTP supports clients which already know their IP addresses. This may sound ridiculous, but remember that it's a bootstrapping, not an address assignment protocol. Dispatching boot images is another task it tackles, and IP address makes a good key to it. That is, the server can use an IP address as an identifier to send different boot images to different clients.

If you think about BOOTP carefully, you'll quickly come across a "chicken or the egg?" sort of problem: how does a client send a UDP datagram to the server if he doesn't know his IP address yet? And how is it expected to receive the reply? The answer is broadcasting. When a client crafts the request, it sets its own address to zero and the server IP address to 255.255.255.255. The latter is a special address that all hosts respond to. Of course, if the client has its server IP address pre-configured, it may use it instead. The server won't be able to resolve the client IP address via ARP unless the client already knows it. However, the server knows the client's MAC address, so it can add a corresponding entry to the ARP table "manually". Broadcasting is also an option. Clients also use randomly assigned identifiers to match responses to requests. When a client receives BOOTREPLY, it installs the IP address it contains and continues as normal.



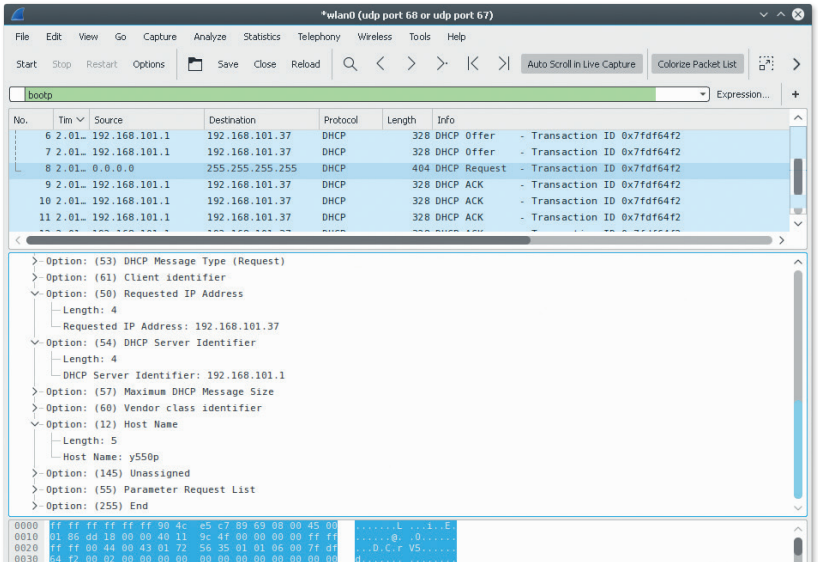
The "DHCP Offer" message provides a client with a range of IP settings the server can assign him.

Modern DHCP

BOOTP already handles centralized IP address management, yet it is rather static. The assumption is hardware addresses map to IP addresses 1:1. This is not enforced, and the server is free to take addresses out of some pool. However, the protocol provides no way to tell the server that the client is not interested in his address anymore. So, sooner or later this pool will be exhausted.

In other words, a dynamic address assignment protocol should only "lend" IP addresses for a limited time. When it elapses, the client must renew its lease. Otherwise, the address is considered free. Of course, a client may terminate his lease prematurely – this is useful if the system is going to shut down, for instance. The procedure is in principle no different from what you normally do in a car rental service.

The protocol that implements it is called Dynamic Host Configuration Protocol, or DHCP (RFC 2131). It's ubiquitous, and when the network "just works" for you, DHCP is what makes it happen. DHCP and BOOTP run on the same UDP ports and use the same message layout. You may think of BOOTP as a "stripped-down" DHCP, for the purposes of bootstrapping only. This is like TFTP relates to "full-fledged" FTP. DHCP is an



The "DHCP Ack" message is an acknowledgement that the server has committed IP settings for the client.

extension to BOOTP, and most DHCP servers are compatible with BOOTP clients.

DHCP introduces several new message types. When a client wants to allocate an IP address, it broadcasts the DHCPDISCOVER message to discover DHCP servers nearby. Each server receiving this message replies with DHCPOFFER, containing an IP address along with other IP settings. At this point, the address isn't assigned yet. The client may receive multiple DHCP offers. It chooses one and replies with a DHCPREQUEST message to the originating server. This way, all other servers know their offer was rejected.

Now the server commits the configuration and replies with DHCPACK. If things don't go well, a negative response (DHCPNACK) is sent, and the procedure restarts. Upon receiving DHCPACK, the client also does its final checks and replies with DHCPDECLINE if it detects any problems.

Otherwise, the address is considered assigned. The lease time is sent along with other DHCP options (see below) in DHCPACK. When this time is about to elapse, the client sends another DHCPREQUEST to renew the lease. If the client decides to relinquish the lease, a DHCPRELEASE message is sent instead.

Dynamic nature of DHCP doesn't mean the addresses themselves must be variable. Most DHCP servers support so-called "static leases", which can assign a permanent IP address to specific hosts. This could be useful, say, for your home server or NAS.

A few parameters are usually required to connect your PC to the internet. Besides an IP address, you need to know your local network mask. You'd want the gateway address to route traffic outside the local network, and at least one DNS server to resolve domain names (LV024). In some cases, you'd also need a web proxy or some custom routes.

In BOOTP, this sort of configuration was conveyed via Vendor Extensions, which become Options in DHCP. The reason is perhaps they are not vendor-specific anymore. DHCP Options are standard. Moreover, one of them is always present and carries the message type we've just spoken about. IP address lease time is also a DHCP Option.

You can distinguish DHCP Options by their numbers. Say, Option 1 sets a subnet mask, while Option 3 stores the list of routers, in order of preference. DHCP Option 6 is for the DNS server. Web Proxy Auto Discovery (WPAD) works via DHCP Option 252, which contains a URL of the Proxy Auto Configuration (PAC) script.

DHCP Options look nice and convenient, but remember that clients may ignore settings that you push to them this way. For tricky stuff, like WPAD, this is rather possible. Better treat DHCP Options as recommendations. If you need these settings enforced, you want something else.

Now, look at two *Wireshark* screenshots we have here. They capture a DHCP client address assignment procedure I triggered with **dhcpcd** (see below). All messages share a Transaction ID because they belong to a single session. Unsurprisingly, everything starts with the "DHCP Discover" message sent to 255.255.255.255, an IPv4 broadcast address. The server replies with "DHCP Offer" sent to yet-unknown to the client unicast address, 192.168.101.37. This works, as the server already knows the client's hardware address, and no ARP request is needed. The lease time is two hours. The client responds with

IP configuration, the DIY way

Just because you can have your IP address assigned automatically doesn't mean you need to. Good old static network configuration still works in 2016.

Linux provides several approaches to manual network configuration. The one is via **ifconfig** and **route**. Another is using the single **ip** tool, which is more versatile.

The roadmap is as follows. You bring the device online, assign it an IP address, then add some routes (at least, the default one) and configure DNS. Imagine you want to configure the network on **eth0** device. Then start with the following (as **root**):

```
# ip link set up dev eth0
```

This brings the device "up", that is, to the active state. Now, assign it an IPv4 address with:

```
# ip addr add 192.168.1.5/24 dev eth0
```

It is up to you to ensure that the IP address you've chosen is not conflicting with anything else on your network. You may also notice a CIDR notation, **addr/mask**. It allows you to assign both the IP address and the netmask in a single command. **mask** is the number of bits set in the network mask, so **/24** is equivalent to `255.255.255.0` in **ifconfig**'s parlance.

Next step is adding the default route:

```
# ip route add default via 192.168.1.1
```

Any valid destination (host or network) is acceptable where **default** is. You need to substitute 192.168.1.1 with your gateway address, which needs to be on the same subnet as you are. Your final step is to tell Linux the DNS server to use:

```
# echo 'nameserver 8.8.8.8' > /etc/resolv.conf
```

Things are trickier if your system uses **resolvconf** or similar management framework. In this case, refer to the man pages.

"DHCP Request", asking the server to commit the suggested configuration. "DHCP Ack" completes the sequence.

Server side

There are many good DHCP servers available for Linux. A *de-facto* standard is **dhcpd** from the Internet Systems Consortium (ISC). For smaller networks, there is often a simpler alternative called **dnsmasq** (**http://www.thekelleys.org.uk/dnsmasq/doc.html**). You may already run **dnsmasq** without being aware: it comes pre-loaded on many home wireless routers.

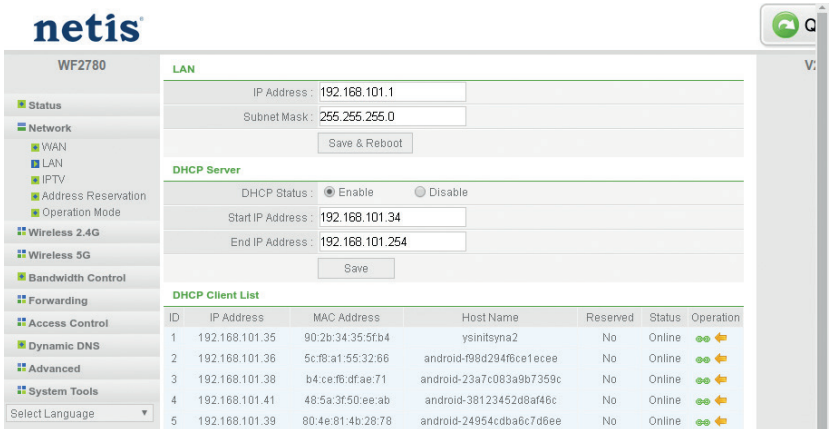
Strictly speaking, **dnsmasq** isn't a DHCP server. It is an all-in-one solution, providing caching DNS server, DHCP, network booting and everything else you may want for you home or small office network. Of course, it's free, and you'll probably find it in your distribution's package manager. **dnsmasq** uses a single configuration file. Typically, it's **/etc/dnsmasq.conf**, and it comes with a lot of comments. The parameters we are interested in start with the **dhcp-range=** line. This enables the DHCP server and supplies it with a pool to assign addresses from. Optionally, you may also specify the lease time here:

```
dhcp-range=192.168.0.50,192.168.0.150,12h
```

That's it – now your clients will get dynamic addresses from the supplied range. Of course, you can do some fine-tuning, if you want to. Say, this is how you assign a given client a static IP address along with the hostname:

```
dhcp-host=11:22:33:44:55:66,fred,192.168.0.60
```

Again, you may append a comma-separated lease time. It makes sense to set it higher than for ordinary clients, as there is no point in re-acquiring the same IP address every few minutes. **dnsmasq** even supports `infinite` lease time, if you want to make the assignment permanent:



You'll often find **dnsmasq** (or **dhcpd**) running behind the facades of modern web browsers.

```
dhcp-host=bert,192.168.0.70,infinite
```

This also shows you may use client-supplied names as identifiers instead of their hardware addresses.

It is also completely possible to set the DHCP options you want. **dnsmasq** already fills some of them with sane defaults. For instance, the daemon assumes that the default gateway and DNS servers are on the same machine as itself. If it's not the case, the following should help:

```
dhcp-option=3,192.168.0.254
dhcp-option=6,8.8.8.8
```

An alternative syntax uses symbolic names, like so:

```
dhcp-option=option:router,192.168.0.254
dhcp-option=option:dns-server,8.8.8.8
```

Now, run **dnsmasq** with an init script, systemd unit or whatever else your Linux uses, and start serving your clients. Note that it is generally not a good idea to run two DHCP servers on the same subnet if they have their address pools intersecting. As you likely already have DHCP enabled on your router, experiment carefully. In any case, if your network starts to misbehave, just tear down **dnsmasq**.

Command of the month: dhcpcd

ISC's **dhcpd** is a *de-facto* standard DHCP server. No wonder its counterpart, **dhclient**, is the *de-facto* standard DHCP client. But Linux is all about choice, and there is one other thing: **dhcpcd**.

Personally, I tend to use **dhcpcd** whenever I want to initiate a DHCP procedure manually. It's simple: you just type **dcpcd** and tell it which interface you want to configure:

```
# sudo dhcpcd wlan0
DUID 00:01:00:01:1f:13:91:56:90:4c:e5:c7:89:69
wlan0: IAID e5:c7:89:69
wlan0: soliciting a DHCP lease
wlan0: offered 192.168.101.44 from 192.168.101.1
wlan0: ignoring offer of 192.168.101.44 from 192.168.101.1
...
wlan0: leased 192.168.101.44 for 7200 seconds
wlan0: adding route to 192.168.101.0/24
wlan0: adding default route via 192.168.101.1
forked to background, child pid 2069
...
```

dhcpcd is quite chatty by default, at it tells you which offers it got and what it decided to do about them. When the process finishes, the command forks into the background and keeps running silently to renew the lease when the time comes. The command also accepts numerous arguments. For instance, you can send arbitrary DHCP options with **-v**. When you decide you no longer want to use your address, run **dhcpcd -k** to relinquish the lease.

Tools like **dhcpcd** are rarely useful in modern desktop environments with their graphical network setup tools. However, they may come handy in disaster recovery systems, or otherwise less user-friendly Linux distributions. It's good to be aware of them just in case you'll need to negotiate a DHCP server manually someday.

LINUX INSIDE:
ROBONAUT R2

Robonaut R2 is a humanoid robot that lives on the International Space Station. Initially designed as an earth-based prototype, R2 proved so capable that it was launched into space on 24 February 2011 on what was the Space Shuttle Discovery's final mission. It is, at the time of writing, the only humanoid robot in space. The advantage of this form-factor is that it can interact with parts of the space craft that have been designed to be used by humans.

The brains of the machine are actually located where its stomach would be if it had one. 38 Power PC processors controlled by a Linux kernel control the movement and communications. Users either on earth or in space can operate Robonaught by wearing a visor, vest and pressure-sensitive gloves.

Initially, R2 was just a torso, but legs have been added to enable it to move more easily and NASA even hope that in the future it will be able to perform space walks more effectively that human astronauts.

Nick Veitch was the original editor of Linux Format, a role he played until he got bored and went to work at Canonical instead. Splitter!

I learnt to program on the ZX80. I don't remember how long it took to give up on ZX Basic – do remember trying to convert listings printed for other computers to ZX Basic, but ultimately, it wasn't enough space to realistically do much in Basic, So I eventually had to work out how to use Z80 assembly, which with no room for any proper tools, resulted in a lot of writing things out by hand, converting instructions into numbers and writing a Basic program to 'Poke' the values into memory.

It was a tiresome and error-prone process. I always remember thinking "they said this computer was more powerful than the ones used in the US space program, I bet they didn't write their code like this". Well...

Recently great chunks of the original code for many of the Apollo missions systems has been uploaded to GitHub (**https://goo.gl/p5pccF**). Aside from just the curiosity of leafing through the annotated source, it is amazing to see how tightly coupled this code was with the hardware – every cycle needed to be accounted for, and needed to be resilient to reboots (which happened frequently in some missions) and the software implemented a very basic form of multitasking so that important things (firing the engines) always happened on time irrespective of what other processes were running. The available storage memory (actually, wire-wound ferrite core storage) was so small that many spaces had to be used for multiple variables depending on the phase of the mission – some slots were used for up to seven different variables.

It is a remarkable achievement (and if you are curious you can also find online emulators for the different computers and the DSKEY interface used by astronauts (**www.ibiblio.org/apollo**).

LINUXVOICE

This is what we've done in the last 24 issues. Subscribe to the next 12 from just £38.

Every subscription includes access to every PDF, ePub and audio edition we've ever published.

shop.linuxvoice.com


LV