PROUDLY INDEPENDENT SINCE 2013

LINUXVOICE

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

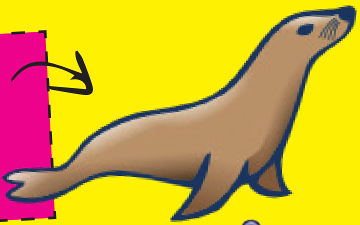
SCRIPTING LANGUAGES

JavaScript Python, Perl – which is right for you?

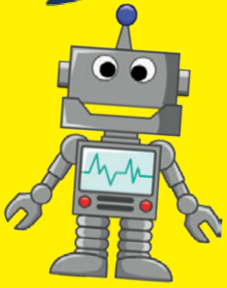
ENCRYPT EVERYTHING

The complete guide to keeping your data private – no matter who's looking

CONTROL A MARIADB DATABASE



OPERATION
Build a practice cadaver for home surgery fun



THE SNOOPER'S CHARTER
Why it's an utterly rubbish idea

YOUR BEST DESKTOP EYE CANDY p22

31 PAGES OF TUTORIALS

JOIN US NOW
LESLIE HAWTHORN
On community, human nature, leadership and the importance of failing properly

FIAT LUX
PHILIPS HUE
Control the lighting in your hollowed-out volcano base with Linux and Python

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

FOG › VLC › SQUID › WINE › TAILS › SED & MORE!

COMING SOON*

FROM THE TEAM AT LINUX VOICE

Beep Beep Yarr!

An adventure story for young minds – featuring pirates, robots and computer programming.

Order from 25 Jan on Kickstarter!
www.linuxvoice.com/book

*With your help!

TWO YEARS OF AWESOME

The March issue

GRAHAM MORRISON

A free software advocate and writer since the late 1990s, Graham is a lapsed KDE contributor and author of the Meeq MIDI step sequencer.

This is our 24th issue. That means it's been two years since we launched, exactly as promised, after a successful crowdfunding campaign. It's a cliché and too often repeated, but we couldn't have done this without you. Without your backing we simply wouldn't be here. I know of no other magazine, anywhere, that has been able to do this, let alone one that gives old issues away for free (even for commercial use!) and shares in any profits.

These are ideas that run totally against traditional publishing wisdom, and once again, it's the Free Software, Linux and open source communities that have made the impossible possible. It's you that has made this possible. So, thank you. All of us here at Linux Voice are incredibly grateful.

Graham Morrison
Editor, Linux Voice

(ps we're attempting to crowdfund a book that will teach children programming via pirates and robots – see our advert on the left.

THE LINUX VOICE TEAM

Editor Graham Morrison
graham@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Technical editor Ben Everard
ben@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, Marco Fioretti, Juliet Kemp, Vincent Mealing, Simon Phipps, Les Pounder, Mayank Sharma, 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#024

ANDREW GREGORY

Most of us don't have the time to go through bureaucratic edicts to check their sanity. But this is what Ben's done with the UK's 'Snoopers' Charter' and the ineptitude he's found is stagging.
p28

BEN EVERARD

As I'm in the process of automating my home, our tutorial on controlling Philips Hue lightbulbs with Linux comes at exactly the right time. Forget Netflix and chill – UNIX and chill!
p80

MIKE SAUNDERS

I loved playing the board game 'Operation' when I was a lad, and it's great to see our own DIY Operation game is way better than the modern version of the same game. Upgrades!
p72

SUBSCRIBE ON PAGE 56

Contents

Welcome to the 24th issue of Linux Voice. Blimey, two years eh?

NEWSANALYSIS

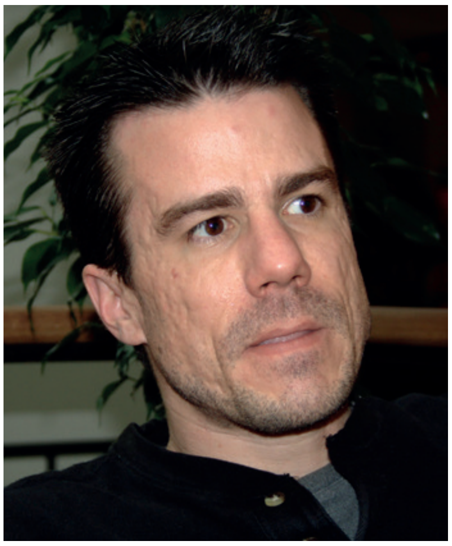
The Linux Voice view on what's going on in the world of Free Software.

Opinion

Remembering Ian

Without Ian Murdock, Linux would be nowhere near the success story it is today.

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.



Along with my former colleagues at Sun Microsystems, I was horrified at the start of the year to hear of the death of Ian Murdock. There have been plenty of other comments written about him, so you will be aware of his work establishing Debian, and maybe of his own company (Progeny) and his work at what would become the Linux Foundation. While you'll remember Ian as the spirit and name behind the Debian project, I got to know him in 2004 when he joined Sun Microsystems. I was the head of open source activities at Sun, while Ian was hired a little later to head up the transformation of Solaris into an open source operating system.

Linux for the people

The outcomes from each of our jobs are stories for another time, but I was struck by the quiet force Ian brought to his role. Amid a good deal of scepticism and hostility – after all, Ian's creation was a driving force behind the demise of Solaris – Ian was able to shape the work that the OpenSolaris team was doing, driving them in directions far more likely to result in community engagement than the instincts and directions of their previous management. He showed determination and vision, and used his experience at the convergence point of business and technology to talk round both the engineering genius and the pointy-headed boss.

The key vector from Debian that was introduced to OpenSolaris was the need for the software to be delivered in the form of installable packages from an open repository, rather than the previous vision of a tightly-controlled single source for componentry.

The importance of packages

That vision of an operating system built from packages with their dependencies automatically resolved and with the legal and social issues related to each element discussed and addressed in advance is the key legacy Ian leaves us.

We take it for granted today that we can simply add a package to our operating system with a single command and without destroying the system or spending the rest of the day resolving dependencies and incompatibilities, and we are extending that concept further into containers and cloud deployment. But when Ian started his Linux distribution, that was not the received wisdom, and I credit him with proving it was crucial and showing how to lead a community to make it real.

More than that, Ian was also a great friend to many of us. While his history with Linux may have aroused suspicions in the team at a traditionally proprietary software vendor, we all quickly realised that Ian was a warm, kind and complex person who we enjoyed being with.

To lose him so early is a tragedy; while I have the greatest respect for his technical legacy, what I will miss in the future is the opportunity for his friendship.

> We take it for granted today that we can simply add a package to our operating system with a single command

Kernel • Swift • IPv6 • LibreOffice • Ian Murdock • PlayStation 4

CATCHUP

Summarised: the biggest news stories from the last month

1 Linux kernel 4.4 released

To ring in the new year, we have a shiny new kernel release. Linus Torvalds announced kernel 4.4 on 10 January 2016 with a boatload of new features: support for direct I/O and asynchronous I/O in the loop block device, 3D support in the virtual GPU driver (for hardware accelerated graphics in virtualisation guests), support for open-channel SSDs, and many improvements in drivers, filesystems and memory management. See here for details: **http://kernelnewbies.org/Linux_4.4**

2 Debian Founder Ian Murdock passes away

Just after Christmas 2015, Ian Murdock, founder of the Debian GNU/Linux distribution, died in San Francisco at the age of 42. The cause of death is as yet unknown: hours before he had been threatening suicide on Twitter after alleged mistreatment by police forces. According to the police, he had been detailed for attempting to break in to an apartment while drunk. It's a sad loss for the Linux community – for a look back on Ian's work, see Simon's words left and p9.

3 PlayStation 4 gets hacked to run Linux

It's been a while coming, but finally an exploit for Sony's console has been developed to enable low-level access to the hardware, and therefore the ability to run Linux on the machine. **http://tinyurl.com/zp3g5bn**

4 Dutch government supports encryption

At at time when many governments are eager to sneak backdoors into encryption software, ostensibly to "protect us", the Dutch are taking a different approach. The powers that be there have stated that "it is currently not appropriate to adopt restrictive legal measures against the development, availability and use of encryption within the Netherlands". And on top of that, the government has approved a €500,000 grant to the OpenSSL project.

5 IPv6 celebrates 20th birthday with 10% usage

Version 4 of the internet protocol has done us well over the decades, but its mere 4.3 billion IP addresses is becoming a severe limitation with the Internet of Things coming into full flow. IPv6 offers vastly more addresses (and many other features) but has seen slow adoption. Still, as of December 2015 it is now 20 years old and has reached 10% adoption according to Google. Is this the start of a widespread switchover? The next 12 months will tell...

6 Apple's Swift language comes to Linux

Originally unveiled at Apple's 2014 Worldwide Developer Conference, Swift is a programming language created for iOS, OS X and the Apple Watch. It was designed to be more concise and safe than Objective C, which the company had been using for many years. Now the language is open source, and can be used to develop software on Linux as well as Apple's proprietary platforms: **https://developer.apple.com/swift/blog/?id=34**

7 Mozilla stops working on Firefox OS smartphones

This isn't entirely unsurprising news, given how utterly dominant iOS and Android are in the mobile space, but it's a shame nonetheless. The Mozilla Foundation has announced that it will stop developing and selling smartphones with Firefox OS, leaving the fledgling platform a possible future on smart TVs and other devices. Which poses the question: is there room in the smartphone market for another OS? Can anyone else break the duopoly that iOS and Android enjoy?

8 LibreOffice Online joins up with OwnCloud

Collabora, the company working on an online version of the *LibreOffice* suite, has teamed up with OwnCloud to create a virtual machine test image that combines both projects. This lets you share, sync and manage your files in OwnCloud, and also edit them from your web browser in *LibreOffice*. It's early days, but it shows great potential and could be a big challenger to Microsoft Office 365. **https://owncloud.org/blog/libreoffice-online-has-arrived-in-owncloud/**

DISTROHOPPER

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

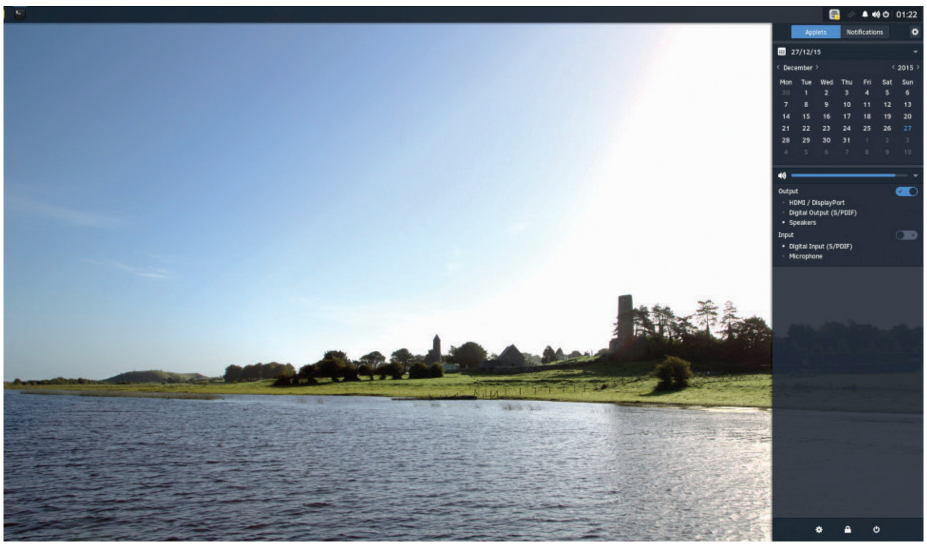
Solus 1.0

Featuring the Budgie desktop.

Most "new" Linux distributions that come onto the scene are based on other distros – standing on the shoulders of giants, if you will. This makes a lot of sense in most cases, but sometimes it's good to start completely from scratch and build a distro from the ground up. That's what the developers of Solus have done, and now they've finally got version 1.0 out of the door.

Codenamed "Shannon" (after the longest river in Ireland), Solus 1.0 aims to be an attractive and user-friendly desktop distro that's focused on the x86-64 architecture. One of Solus' most notable features is Budgie, a custom desktop environment that's built on top of *GTK* and Gnome. Budgie is designed with accessibility in mind (especially for visually impaired users).

One feature unique to Solus is its package manager, called *eopkg*. This supports the usual functionality of adding, removing and searching for packages, and these packages are neatly organised into categories that show what they provide).



Raven (on the right-hand side) is the Budgie desktop's panel containing applets and notifications.

Given that there are so many top-quality Linux distros out there, we're often sceptical when we see new ones being created from scratch, as they tend to exhibit a lot of "not invented here" syndrome – ie being built for the sake of it, rather than to fix any particular problem. But the Solus team have a solid vision for their distro, and have paid plenty of attention to presentation and marketing, something lacking from so many smaller-name projects.

We wish the project luck. To try out Solus and read more about it, visit the distro's main website at **www.solus-project.com**.

OpenSUSE Li-f-e 42.1

Education distro with long-term support.

Release numbering and naming in the Linux distro world is often... creative, to put it mildly. Not only did OpenSUSE recently make the giant leap from version 13.2 to 42.1, now a bunch of developers have created a new spin-off focused on education and schools. OpenSUSE Li-f-e 42.1 (the Li-f-e means "Linux for education") is described as "the only enterprise-grade long-term supported Linux distribution for education", and it's a live DVD/USB image that can also be installed to a hard drive.

What makes Li-f-e shine for education use is the software selection. Li-f-e includes a range of packages designed for learning, such as the *Parley* vocabulary trainer and *Marble* virtual globe.

Some of the other education-oriented packages include *GCompris*, a set of learning games geared towards young children, and *Little Wizard*, a kid-friendly introduction to the concepts of programming. Then there's *TuxPaint*, *iGNUit* (a flash card tool), *gElemental* (a periodic table viewer) and *Stellarium* (for exploring the stars).

Why should schools pay for Windows when there are better, cheaper alternatives?

Linux distros such as Li-f-e are much safer than Windows and provide boatloads of great software for kids out of the box, so we welcome attempts by the major distro vendors to get involved. To find out more and download the release, check out **https://lizards.opensuse.org/2015/12/21/announcing-li-f-e-42-1**.

News from the *BSD camps

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

How small can a BSD be? Well, if you take the (rather ancient) source code to 2.11 BSD, update it for modern compilers and hardware, and squeeze it down a bit more, you can get it running on a device with just 128k of RAM. That's a full BSD operating system with memory protection, multitasking and POSIX compatibility. All of this is thanks to the work of the RetroBSD project (**www.retrobsd.org**), which is targeting microcontrollers such as the Microchip PIC32.

This might seem like a rather pointless task in this day and age, but it's quite the opposite – consider how much work is going into embedded devices, especially as the much-lauded "Internet of Things" starts to take off.

Meanwhile, DragonFly BSD saw a new release in December: 4.4. For those who've never heard of it before, DragonFly is a fork of FreeBSD that came to life in June 2003 after a fall-out among developers. Its chief coder is Matt Dillon, who some long-time hackers may remember from his work on the Amiga (such as the *DICE* C compiler). With DragonFly BSD, Dillon wanted to take a different approach to FreeBSD in dealing with some key architectural issues, such as threading and symmetric multiprocessing (SMP).

Although DragonFly is still a much smaller project than FreeBSD, it has pioneered plenty of technologies such as the 64-bit HAMMER filesystem featuring infinite NFS-exportable snapshots, configurable history retention, and checksums to ensure data integrity. Version 4.4 of the operating system includes better support for Intel and Radeon graphics (thanks to drivers from the Linux kernel), improved CPU power saving settings, and a vastly enhanced regular expression library. For the full release notes and links to downloads, visit the project's website: **www.dragonflybsd.org/release44**.

While DragonFly BSD is often overshadowed by FreeBSD, it has plenty of unique features to warrant trying it out.

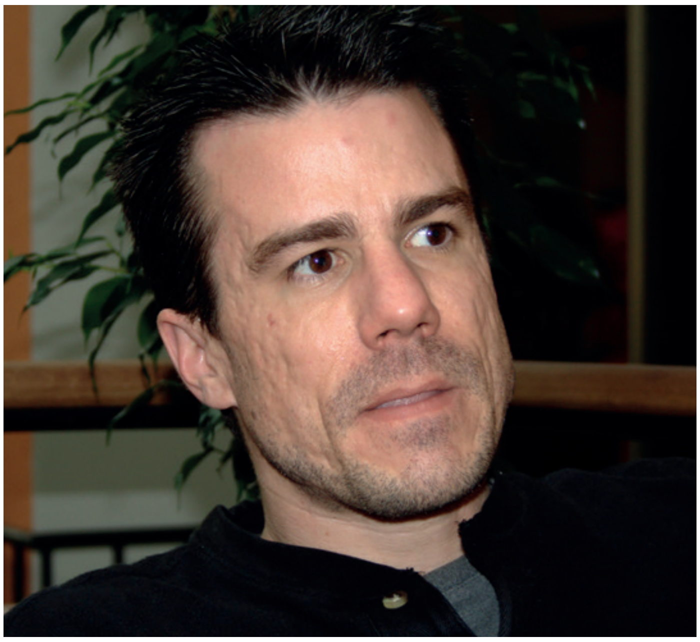
In memory of Ian Murdock (1973–2015)

Anyone can start a Linux distribution – but few can make it a long-term success. Back in the early 1990s, as the GNU/Linux combination was still very much a baby, the only 'distributions' around were small, hobbyist hackish projects for getting the operating system onto your hard drive one way or the other. But Purdue University student Ian Murdock saw the chance to create a new kind of distro: one that focused on the community and spirit of Free Software. As he wrote in the Debian Manifesto in 1993:

"Debian Linux is a brand-new kind of Linux distribution. Rather than being developed by one isolated individual or group, as other distributions of Linux have been developed in the past, Debian is being developed openly in the spirit of Linux and GNU. The primary purpose of the Debian project is to finally create a distribution that lives up to the Linux name. Debian is being carefully and conscientiously put together and will be maintained and supported with similar care."

Debian was named after Ian and his then-partner Debra, and the distribution picked up support from the Free Software Foundation early on. Since then it has become an enormous success, not just as a distro in its own right, but as the basis for Ubuntu, Mint and many other derivatives. Debian's consistent and stringent focus on democracy, community, engineering and freedom has produced a distro that millions now rely on for their day-to-day work.

So thanks, Ian, for having the foresight and skills to start such a monumental project. And although everyone will remember you for Debian, thanks for your additional work in the Free Standards Group, Linux Standards Base and Linux Foundation. The world of Free Software is so much stronger due to your efforts.


Ian Murdock started Debian, arguably the most important Linux distro ever (photo: Ilya Schurov, CC-BY-SA, **www.flickr.com/photos/39112057@N00**).

YOUR LETTERS

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

STAR LETTER

HALLOWEEN IS OVER

LV issue 023 contained, as have prior numbers, many jabs at Microsoft as the natural enemy of the Free Software believer. It's time to accept that the world has changed.

Like many among your staff and readers, I remember that period when the infamous Halloween memos were leaked, and we realised joyfully that the Free Software movement was big enough to concern the biggest software company in the world.

I remember this not because it was recent, but because I am old: this happened in 1998. Large companies like Microsoft can be slow to change, so it is right that we remain sceptical of their intentions with Free and open source software, but we need to remember that if we define our movement as Anti-Microsoft, it will live or die by their fortunes alone.

While we jab at Azure for their plush Tux swag, Apple has become one of the largest companies on the planet. It has done this with its proprietary iPhone and iOS platforms, which lock in more first-party applications than 1990s Windows did when the antitrust cases started flying. You can download alternatives from its store (and its store alone), but the terms of business on that store prohibit copyleft software. The downloads obtained by Apple's users are restricted by DRM to particular Apple accounts.

Meanwhile, Apple co-opts open source projects like Clang and LLVM to replace successful Free Software components like GCC. How does the availability of a cuddly Tux with Microsoft branding stack up to these actions in respect to the FSF's four freedoms?

We celebrate Google for popularising the Linux kernel through its Android mobile OS, and companies like it, including Facebook and Twitter, for their contributions to open source software. However, these companies thrive by providing proprietary services from their own server farms. None has embraced the AGPL, a licence that extends freedom to remote users of a hosted service [and which Mike talks about in this issue's Group Test on page 50]. Is it meaningful to have the freedom to use a browser or a mobile device for any purpose, if the available purposes involve using non-free services?

So yes, Microsoft is still important, and its proprietary Windows and Office products are still huge obstacles to the freedom of computer users everywhere. On the other hand, Microsoft is no longer the headline company defining the computing landscape for many people. If the Free Software movement is the "say no to Microsoft" movement, then we will not win. Rather we will become irrelevant at the same time as our nemesis in Redmond.

You may think that Steve Jobs is an unlikely role model for someone in my position, but I will end by paraphrasing his statement on his return to Apple. We need to get out of the mindset that for the Four Freedoms to win, Microsoft has to lose.

Graham Lee

Should we move on from our obsession with Microsoft, or is it useful to have such an incompetent enemy?

Andrew says: I had never stopped to consider this, but what you say makes 100% sense. In practice though, for most people Microsoft is still the embodiment of proprietary software. Apple is arguably a more serious threat, but Microsoft keeps shooting itself in the foot, so it's an easier target for us. Apple at least makes a lot of good products along with its egregious attitudes towards compatibility, planned obsolescence and forced upgrades; Microsoft seems to be successful only by abusing its market position.

INTERNET OF FOOD

My idea is to make dumb fridges smarter by creating an inventory management gadget from a Raspberry Pi or Arduino, a barcode scanner and a display.

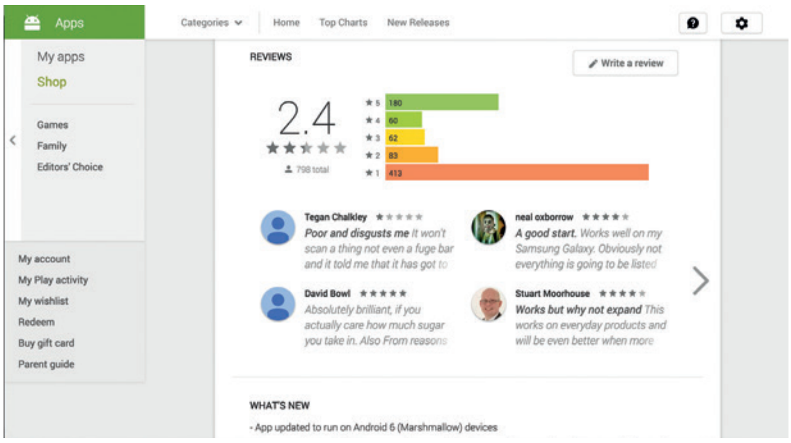
People waste a lot of money because they throw away food that's gone off. I think in most cases they forget about the expiry date of food they put into the fridge. That's why I think a little computer could help out, if it could keep track of expiration dates of various foods and issued alerts before a configured number of days.

I've found a post about this idea, but I think the work was either never done or never published. See **https://www.raspberrypi.org/forums/viewtopic.php?f=41&t=50916**.

So I would like to see a tutorial about something like that in the magazine.

Géza Búza

Andrew says: I would have no interest in this, as all food in my house goes in my belly. However, this project does sound like a far more worthwhile project than the ludicrous sugar app that's recently been developed at UK taxpayers' expense. We'll see if Ben wants to give it a go in his Internet-of-Things robo-house.



REVIEWS

2.4
798 total

★ 5	180
★ 4	60
★ 3	62
★ 2	83
★ 1	413

Tegan Chalkley ★★★★★
Poor and disgusts me It won't scan a thing not even a fuge bar and it told me that it has got to

neal oxborrow ★★★★★
A good start. Works well on my Samsung Galaxy. Obviously not everything is going to be listed

David Bowl ★★★★★
Absolutely brilliant, if you actually care how much sugar you take in. Also From reasons

Stuart Moorhouse ★★★★★
Works but why not expand This works on everyday products and will be even better when more

WHAT'S NEW
- App updated to run on Android 6 (Marshmallow) devices

In the internet of things, your healthcare provider knows how much sugar you're consuming.

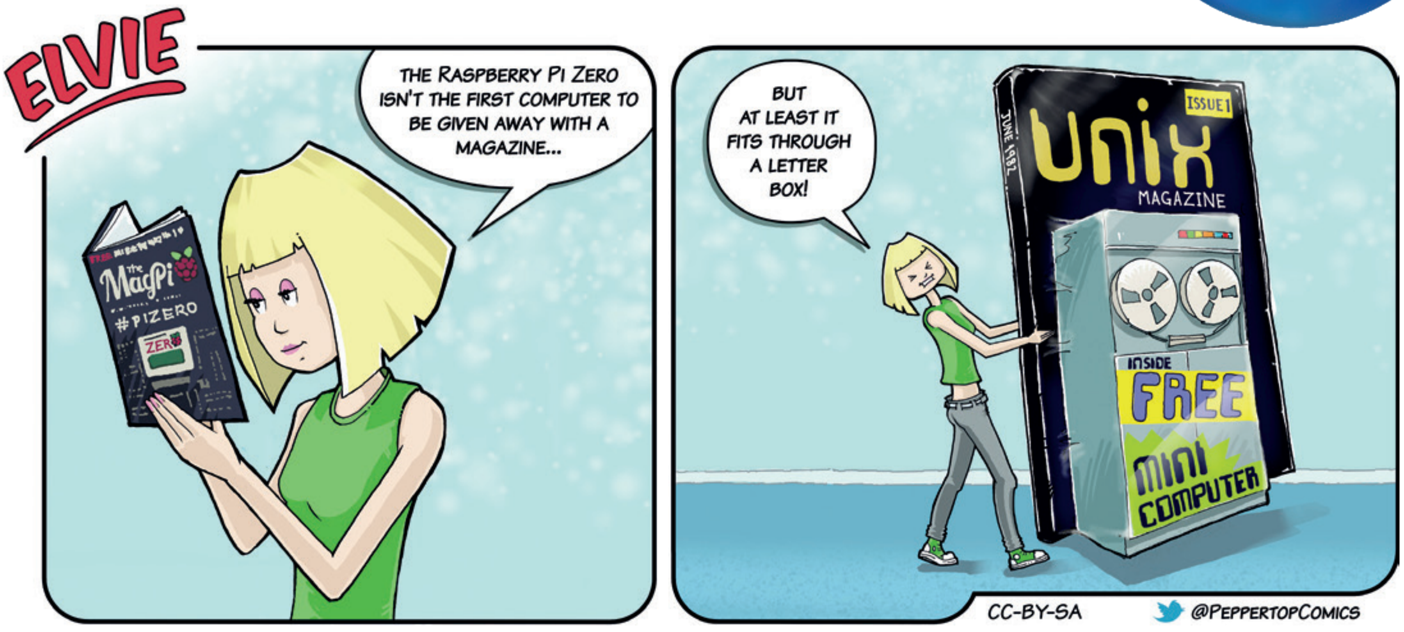
MINIMALISM

I was rather taken aback to see your reference (February 2016 issue, page 8) to Slackware as a "fairly minimalist" distro. Slackware comes with much more out-of-the-box capability that most distros. In addition to a full KDE desktop and application suite, six desktop environments, multiple media players, and almost every text editor known to geekdom, Slackware includes the complete LAMP stack, the OpenSSH server, CUPS server, Samba server, all the libraries needed for compiling from sources, and other functionality that, with most distros, requires obtaining additional packages after the initial install.

Slackware may be many things, but "minimalist" is not one. Let me suggest instead "elegantly simple."

Frank Bell

Graham says: The minimalism to which we referred was to do with the way it treats its packages. Rather than adding bits and bobs, renaming features and fiddling with configuration files (like some distros do to the software in their repositories) Slackware keeps them clean, unadorned and as the developer intended. One might almost say that this approach to packaging is minimalist, but 'elegantly simple' fits the bill much better.


ELVIE
THE RASPBERRY PI ZERO ISN'T THE FIRST COMPUTER TO BE GIVEN AWAY WITH A MAGAZINE...
The MagPi
#PIZERO
ZERO
BUT AT LEAST IT FITS THROUGH A LETTER BOX!
JUNE 1982
ISSUE 1
unix MAGAZINE
INSIDE FREE MINI COMPUTER
CC-BY-SA
@PEPPERTOPCOMICS

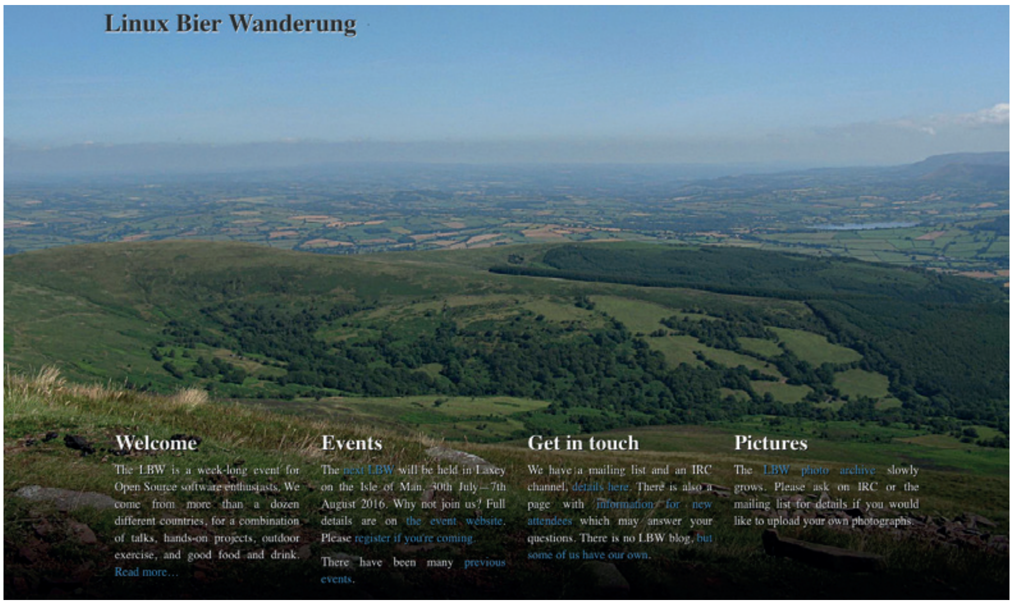
LUGS ON TOUR

The 2015 LinuxBierWanderung

Jason Irwin reports from an event that combines three of our favourite things.

The LinuxBierWanderung is an "unconference" with no fixed fee, just a voluntary donation towards costs. There's no central organisation beyond those who have taken it upon themselves to arrange that year's event. The three components mesh together to provide an event with wide-ranging scope that suits the geek, the family and even the dog. Yes, delegates from various species were in attendance.

Most delegates to this year's LinuxBierWanderung stayed in 'Camping Kaul', which offered excellent facilties and even laid on a Letzebuergian three-course meal on Friday that was simply superb. Camping Kaul also provided a decent Wi-Fi service over the site, this meant there was no need for a rustled-up radio link back to the hall as in previous years.


Linux Bier Wanderung

Welcome

The LBW is a week-long event for Open Source software enthusiasts. We come from more than a dozen different countries, for a combination of talks, hands-on projects, outdoor exercise, and good food and drink. Read more…

Events

The next LBW will be held in Laxey on the Isle of Man, 30th July – 7th August 2016. Why not join us? Full details are on the event website. Please register if you're coming.

There have been many previous events.

Get in touch

We have a mailing list and an IRC channel, details here. There is also a page with information for new attendees which may answer your questions. There is no LBW blog, but some of us have our own.

Pictures

The LBW photo archive slowly grows. Please ask on IRC or the mailing list for details if you would like to upload your own photographs.

Linux

Talks this year covered everything from designing for extreme performance, the legalities and intricacies of professional drone flying, computer vision with a Lego-sorting Lego-bot and all the way to packing for multi-day adventures with less than 5kg of equipment. There were also hands-on sessions with "Microtek" routers (including how to break everything by creating packet storm) and a detailed run-down on how the hall infrastructure had been installed and configured after it had been lifted off the roof of a trusty Land Rover.

Bier

This year's organisers had arranged a hot-line to the local brewery, Simon, which was always prepared to slake the thirst of the delegates. This meant the in-hall bar proved exceptionally popular as not only was the beer of good quality, but there was plenty of it!

Wiltz is a charming town in the Ardennes where, through a quirk of geography, the direction of travel between two points is always uphill.

There was the traditional "Pot Luck" where delegates used the well appointed kitchen to lay on veritable banquet of dishes from their home nations.

Usually a "Pub of the Day" is appointed, but this year it began as more of a "Restaurant of the Day", and over the course of the week delegates mostly elected to simply hang out at the hall to chat, play games and (of course) do their very best to drain Simon.

Below left: Iggy Pup and Alice Pooper contemplate BRAAS (Belly-Rubs As A Service).
Below right: A hexacopter on display – assemble before use.

Wanderung

The final component in this triptych are the walks and other outdoors activities. This year took in the stunning scenery around Lac Haute Sûre, as well as the historic towns of Esch-sur-Sûre and Bastogne.

The 2016 LBW will be held in Laxley, Isle of Man from 30 July to 7 August. See you there!

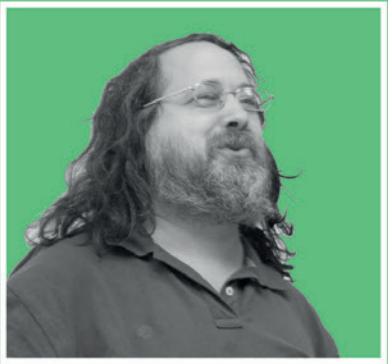
The **Free Software Foundation**, GNU Project and MIT's SIPB invite you to

LIBRE PLANET 2016

"Fork the System"

A conference for everyone who loves free software

Richard Stallman



Fork the System

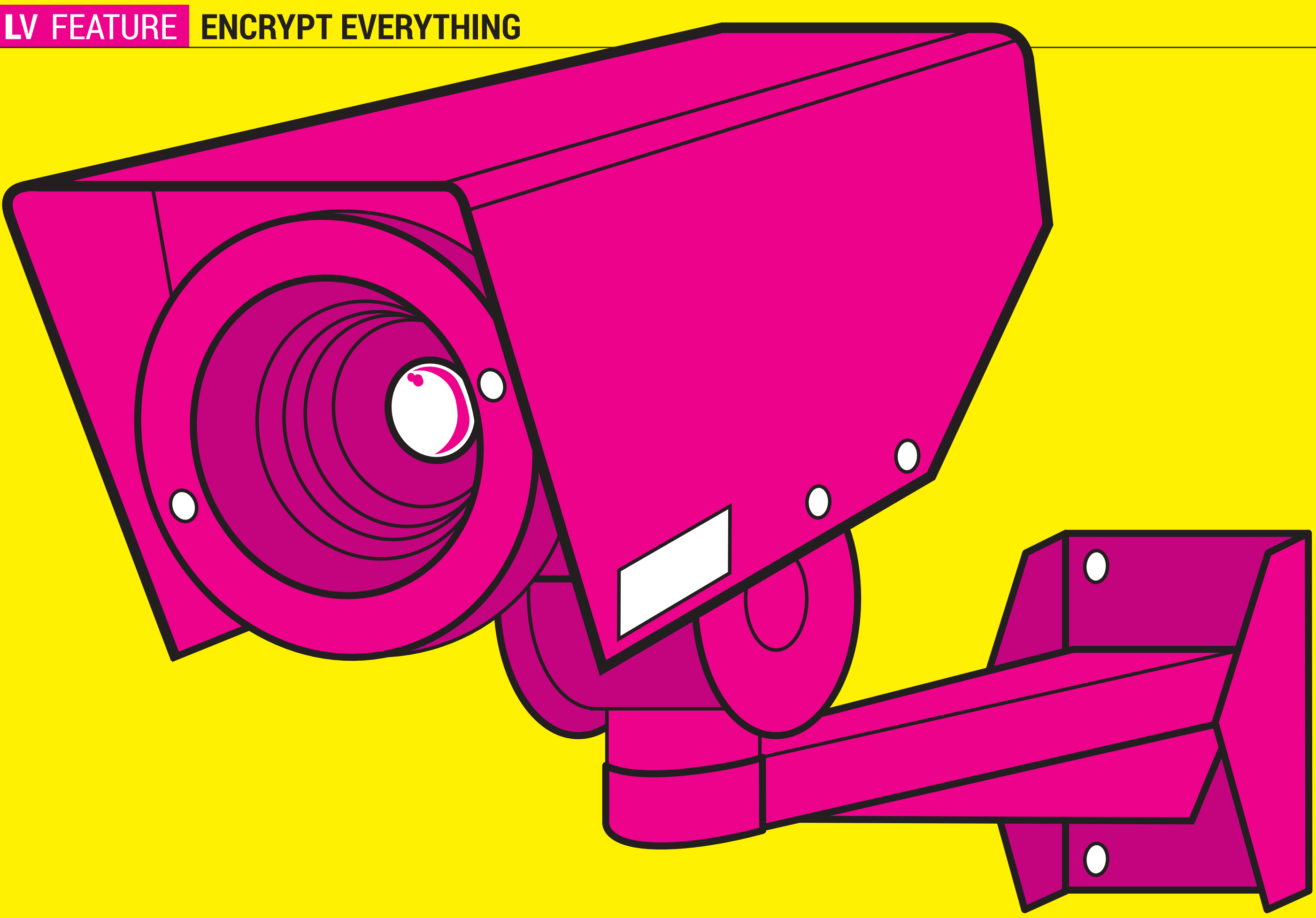
Organized around the theme "Fork the System," we'll examine how free software creates the opportunity of a new path for its users, allows developers to fight the restrictions of a system dominated by proprietary software, and is the foundation of a philosophy of freedom, sharing, and change.

Sessions by

+ Hackers
+ Activists
+ Beginners
+ Users
+ Writers
+ Artists

March 19 & 20 at MIT. Students and FSF members attend gratis.

libreplanet.org/conference



ENCRYPT EVERYTHING

Keep your data secure, safe and private, with a little help from Linux and **Graham Morrison**.

Encryption isn't just important: it's vital. It's as vital as free speech. It enables free speech. Not just for people struggling in places where their voices can't be heard, but for people who want to keep their voices private. It's also vital for security, keeping your data locked when even the data itself is stolen. It's vital for online commerce, banking, mobile phones and digital media, and it's vital for almost everything else we do in the 21st century. Without encryption, we wouldn't have the internet revolution.

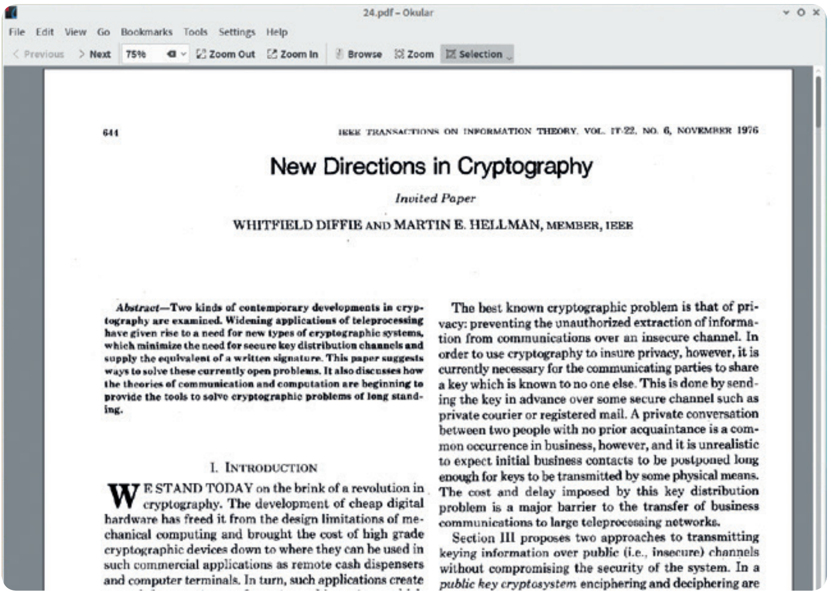
But the only way to trust encryption is through Linux and open source. Without open source there's no way of knowing when software is sending information to a third party, or when a data vulnerability hasn't been patched, or whether the encryption is as strong as its developer says it is. Without open source, there's no peer review, transparency or even accountability. But open source encryption is still a minefield of technology, terminology and complexity, diminishing its effectiveness and veracity. We should all be able to use encryption without any specialist knowledge or training, and that's our target for the this feature – demystifying what encryption is all about so we can all take advantage and put these ideas into practice.

KEY CONCEPTS

Never again get your private keys muddled with your public keys.

There are a couple of reasons for the aura of complexity that hangs over encryption. The first is that the complicated maths behind encryption. But unlike the complex management systems that control your car, for example, in encryption those systems aren't always hidden, exposing the user to ideas and concepts that aren't necessary for its use.

Another reason for this complexity is the difficulty in creating a system that effectively hides these parts while maintaining trust. Without trust, there's no encryption. So, until some clever startup solves the problem of creating easy encryption that everyone can trust without requiring any specific knowledge, it's worth taking a little time to understand some of the key concepts behind encryption, and how you can make it work for you.


24.pdf – Okular
File Edit View Go Bookmarks Tools Settings Help
Previous Next 75% Zoom Out Zoom In Browse Zoom Selection

644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

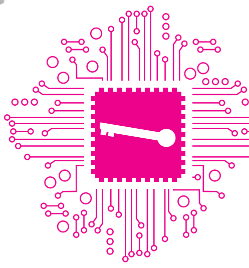
I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are

It was Whitfield Diffie and Martin Hellman's hugely influential 1976 paper on key exchange that led to the proliferation of asymmetric encryption algorithms.



Encryption in three terms

Encryption This is the idea that you can encode something such that it can only be decoded by someone holding a specific decoding key, an idea that has been around for thousands of years. This is the art of cryptography, practised by everyone from the ancient Egyptians to the Third Reich.

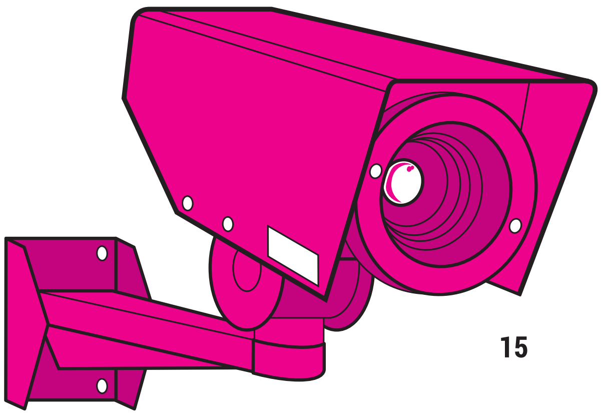
But it's the mathematical prowess of computers that have defined the modern era of cryptography, replacing initiate hieroglyphics or positional rotors with a complex series of substitution and permutation processes (depending on the algorithm) that aim to make the translation of values from 'plaintext' to 'ciphertext' and back to 'plaintext' as difficult and as provably mathematically rigorous as possible.

Algorithm This is the part that turns plaintext into ciphertext and back. The algorithm will depend on how you want your encryption to be used. Mostly, these algorithms are split between two types: asymmetric and symmetric. Asymmetric algorithms are perhaps better known, thanks mostly to SSH and GnuPG/OpenPGP. These use what's known as a private/public key system implemented by the DSA algorithm, although this can be changed. The most important characteristic of this algorithm is that one key can't be derived from the other, and because the private and public keys are different, the algorithm is 'asymmetric'.

Symmetric encryption uses the same key to encode and decode the data. The best known use of a symmetric system is HTTPS, used to secure the web via either the Transport Layer Security or the Secure Sockets Layer. Although a certified public key is used as part of the negotiation process, they keys for the communication itself are generated uniquely for each connection. This is because symmetric encryption is simpler and faster to implement.

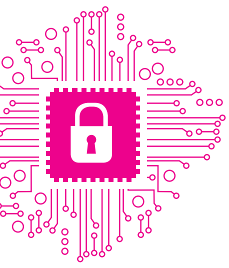
Asymmetric algorithms allow data to be decrypted with a key that's different to the key used to encrypt the data

Public Key/Private Key These terms often cause confusion. It may be because most of us expect encryption to behave the way doors do where the same key is used to both lock and unlock the door. This is the equivalent of a symmetric algorithm (see above), but it's not the only case. Asymmetric algorithms allow data to be decrypted with a key that's different to the key used to encrypt the data, and vice-versa, to use the key to encrypt a message that can only be decrypted by another. Both these keys could be secret, but the revolution came when one of the keys was allowed to be pubic. This meant you didn't need a secret exchange of keys to be able to decrypt a message. You could grab just the public key. And you could use the public key to send a message, or more importantly your own public key, to someone you knew held the private key.



GNU PRIVACY GUARD, AKA GNUPG

Start with the best known technology for chatting as securely as in your living room.



For Linux users and open source enthusiasts, GnuPG is the default choice for nearly all encryption duties. It's capable of both symmetric and asymmetric encryption, and while it's often used to encrypt local data, you'll mostly find it being used to encrypt communication channels.

GnuPG is compliant with OpenPGP, the encryption standard derived from Phil Zimmermann's work on the original PGP. That means GnuPG works with any OpenPG-compliant application – including email clients such as *KMail*, *Evolution* and *Sylpheed*, password managers and chat applications such as *Gabber*, and compatible proprietary software too.

GnuPG support is also built into both Gnome (via its keyring manager, *Seahorse*) and KDE's *KWallet*, which means you may already be using it without realising it.

GnuPG support is built in to Gnome and KDE, so you may be using it already

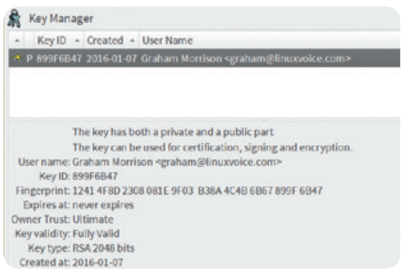
The default way of interacting with GnuPG is via the command line, where the minutiae of every feature and component is best exploited. But the command line can add to the complexity of dealing with an already difficult subject. For that reason, we'd recommend starting your encryption adventure with a GUI front end that enables you to point and click your way through the configuration – that way you'll always have a visual overview of your settings.

Of course, there's a huge variety to choose between, all in various stages of development, using Gnome, KDE and lots of other different environments. But because they're all using GnuPG, and placing their key files and configurations into the same **.gnupg** folder, you'll be able to switch between them without breaking compatibility.

Graphical tools

Both Gnome's *Seahorse* and KDE's *KGpg* are excellent applications, but we'd recommend *GPA*, because it's as close to a default user interface as you can get. Standard applications will detect your configuration and use your keys automatically, asking for your passphrase when needed. Some, including most email clients, will also be able to import a contact's public key. But you can always go back and check *GPA* to see what's been added and whether the configuration is working.

After you've created your secure key pair, you're capable of secure and trusted communication with your contacts. For this to work, you'll also need their public GnuPG/OpenPGP key.



Key Manager

	Key ID	Created	User Name
P	899F6B47	2016-01-07	Graham Morrison <graham@linuxvoice.com>

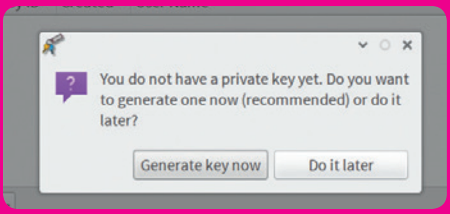
```
               The key has both a private and a public part
               The key can be used for certification, signing and encryption.
   User name: Graham Morrison <graham@linuxvoice.com>
      Key ID: 899F6B47
 Fingerprint: 1241 4F8D 2308 081E 9F03  B38A 4C4B 6B67 899F 6B47
  Expires at: never expires
 Owner Trust: Ultimate
Key validity: Fully Valid
    Key type: RSA 2048 bits
  Created at: 2016-01-07
```

While the command line version of GnuPG is relatively straightforward to use, we'd recommend beginners use a GUI at first.

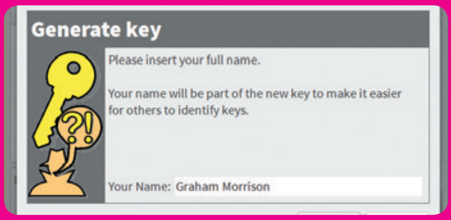
You can do this by meeting in person by accepting a key over a network such as email, OwnCloud, or file transfer.

In order to have ultimate trust in those networked methods, you'll need to verify the key that was received was the key transmitted and from the person you know. This is called a 'web of trust', and while HTTPS solves this with certificates signed by an authority, personal exchanges are different. The simplest way is to check via a phone call, verifying the short fingerprint that accompanies any public key. If the fingerprint is the same then you have the same keys and you can encrypt messages to your contact safe in the knowledge that only they can decrypt the message. Similarly, if you get a message that's signed with the a private key that the public key can decrypt, you can be certain it can only be from your contact.

STEP BY STEP: CREATE YOUR KEY PAIR



You do not have a private key yet. Do you want to generate one now (recommended) or do it later?

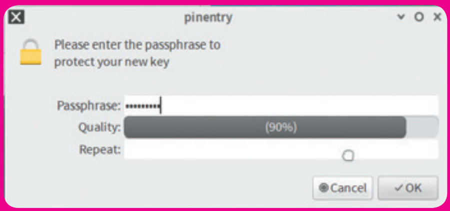


Generate key

Please insert your full name.

Your name will be part of the new key to make it easier for others to identify keys.

Your Name: Graham Morrison



pinentry

Please enter the passphrase to protect your new key

Passphrase: ••••••••

Quality: (90%)

Repeat:

1 Install GPA

The closest thing to a standard GUI for GnuPG is a tool called *GPA* (GNU Privacy Assistant). You need to install this, and when first launched, it will ask whether you need a key pair generating if a pair wasn't already found. Select 'Generate Key Now' to launch the wizard that will create the key pair for you.

2 Enter your details

When asked for your name and email address, you should use your real details. These are used by your contacts to ascertain the validity of your public key. Names and email addresses are obviously not enough, but they're a good reminder. You should also accept the option to create a backup key.

3 Generate the keys

This is where you need to enter the passphrase, and this is more than just a password: not only is it used to encrypt your keys, it's used to unlock your identity. The strength of GnuPG relies on this being unique and unguessable, while at the same time, you need to remember it without writing it down.

EMAIL WITH GNUPG

Public and private keys in practice.

We've seen how important it is to encrypt your data, as well as some of the theories behind how it all works. You'll be reassured to learn that putting these ideas to use is straightforward; but how you do it will depend on your email client of choice.

KDE's *KMail*, for example, integrates GnuPG without any further requirements. All you need to do is associate the key you've just generated with your email account, and you can do this from the Configure dialog by clicking on Identities followed by the Cryptography tab. Use the Change buttons next to the OpenPGP encryption and signing fields to choose your key.

You can change how messages are handled with the Security > Composing panel. Signed messages will glow green if you've got the correspondent's public key in your GnuPG configuration, yellow if not, and you can use the Sign and Encrypt toolbar buttons to sign or encrypt your own emails. When signing or encrypting, you'll be prompted to enter your key's passphrase before the message is sent to your contact.

Encryption add-ons

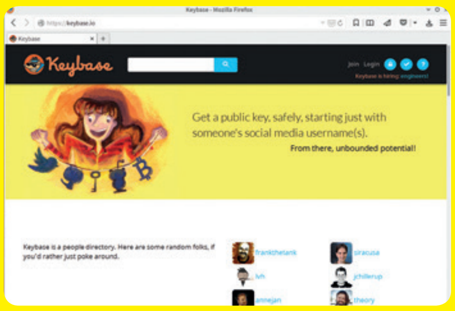
If your email client is Mozilla's *Thunderbird*, you'll first need to install the *Enigmail* add-on via the Settings > Addons menu. With this installed and a restart of the client, you'll be presented with the Enigmail Setup Wizard. Choose the Standard configuration and, thanks to the magic of open standards, the next step should automatically detect your default key pair. Select this and *Thunderbird* is now configured.

When you go back to the inbox view you'll now have the option to encrypt or sign your own emails from the Write Message toolbar, and encrypted messages you receive will be decrypted automatically, just like with *KMail*. Both apps will attempt to download a public key from a key server if you don't have someone's public key in your configuration. This is convenient if you need to read a message, but can't as trusted as a key you've exchanged yourself or through some other method.

keybase.io

As we've mentioned, one of the problems with public keys is being certain the copy you hold is the definitive key of the contact you want to email. Meeting up or speaking via *Skype* is only really viable for contacts you have a relatively close relationship with, which means there's a real problem if you want to send a message encrypted with the public key of someone else you want to email, or if you've received a signed email from them and want to verify the veracity of your sender. One solution is called a key server, a simple repository of public keys linked to email addresses and names. You can publish your own to a keyserver directly from GnuPG's *GPA* tool and most GUI clients, as well as download the public keys of contacts.

But key servers don't solve the fundamental trust problem: you can't be sure the person who uploaded the key is the person you think they are. This is where Keybase (**https://keybase.io**) comes in. It's a key server that works by linking public keys to their respective online personas. If you follow someone on Reddit, Twitter and GitHub, and their accounts are linked via Keybase, you can be almost certain that those keys are for the correct person. There's even a keybase command line tool for extracting keys and sending messages – all you need to import those keys into your system and use them yourself. Adding yourself to Keybase is almost as simple (although it's currently Private Beta). You link your Keybase account to your various networks by posting a Keybase-generated proof to those channels. This is validated automatically and your credentials are added to your account. The final important ingredient works a little like a social network. People who trust your account link their identity to yours, building an all-new peer-to-peer web of trust out of the humble GnuPG. **https://keybase.io**.


Keybase - Mozilla Firefox

https://keybase.io

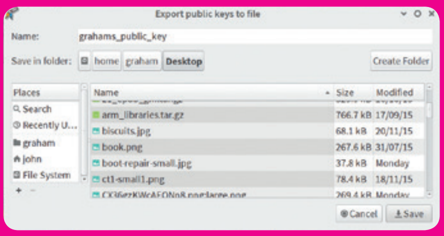
Get a public key, safely, starting just with someone's social media username(s).

From there, unbounded potential!

Keybase is a people directory. Here are some random folks, if you'd rather just poke around.

You can even use Keybase to generate key pairs and encrypt stuff, but we'd still recommend keeping your private key decryption local.

STEP BY STEP: SHARING KEYS


Export public keys to file

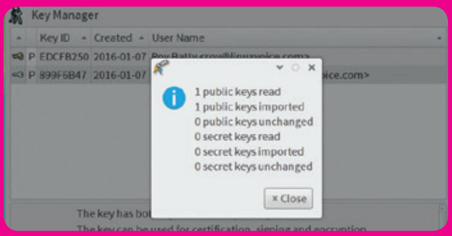
Name: grahams_public_key

Save in folder: home graham Desktop

Name	Size	Modified
arm_libraries.tar.gz	766.7 kB	17/09/15
biscuits.jpg	68.1 kB	20/11/15
book.png	267.6 kB	31/07/15
boot-repair-small.jpg	37.8 kB	Monday
ct1-small1.png	78.4 kB	18/11/15

1 Export your key

One of the best ways of swapping public keys with someone is to swap while you're both physically together. That way you can be certain the key you're getting belongs to who you think it does. To share your key, select Keys > Export Keys from *GPA* and place the resulting file on a USB stick.


Key Manager

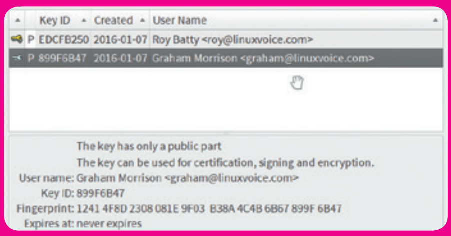
	Key ID	Created	User Name
P	EDCFB250	2016-01-07	Roy Batty <roy@linuxvoice.com>
P	899F6B47	2016-01-07	...ice.com>

1 public keys read
1 public keys imported
0 public keys unchanged
0 secret keys read
0 secret keys imported
0 secret keys unchanged

The key has bot...
The key can be used for certification, signing and encryption.

2 Import a key

Your contact will have done the opposite, exporting their public key to device from which you can now import it using Import Keys from the Keys menu. A status pane will inform you if they key was imported successfully. You can always import more than one, if you're at a gathering where keys are typically shared.



	Key ID	Created	User Name
P	EDCFB250	2016-01-07	Roy Batty <roy@linuxvoice.com>
P	899F6B47	2016-01-07	Graham Morrison <graham@linuxvoice.com>

```
             The key has only a public part
             The key can be used for certification, signing and encryption.
  User name: Graham Morrison <graham@linuxvoice.com>
     Key ID: 899F6B47
Fingerprint: 1241 4F8D 2308 081E 9F03  B38A 4C4B 6B67 899F 6B47
 Expires at: never expires
```

3 Viewing a key

With the third-party key imported, it will be listed within *GPA* along with some information about the contact, such as the encryption used and their key's fingerprint. This is a 'hash' value for the entire key that's often used a shortcut to quickly check whether someone's key is valid or has changed.

SECURING YOUR DATA

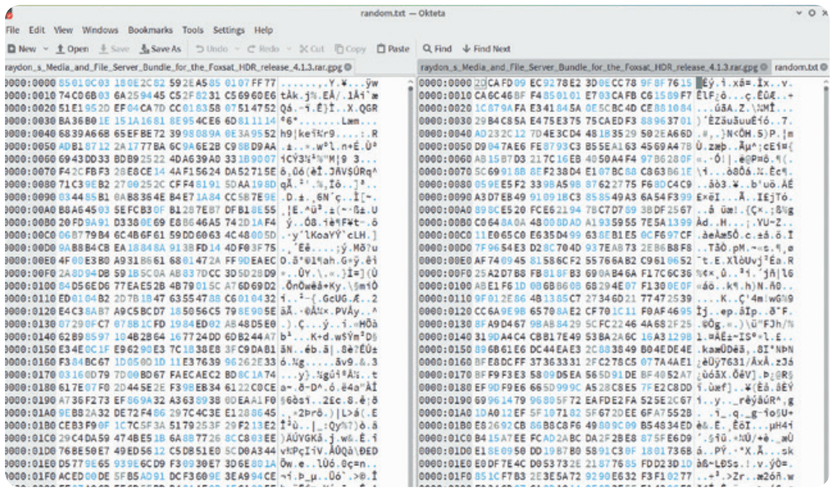
Even if you're not sending files across the internet, it's worth keeping your data safe.

There are many tools created specifically for encrypting your data, and one of the best things you should do is take advantage of your distribution's installer to encrypt your home folder/partition. Ubuntu, in particular, has a great system for encrypting your home folder using eCryptfs (see next page). That way, if you ever lose your laptop, your data will remain secure. But it's also reasonable to take a more *ad-hoc* approach, only encrypting those files or folders you want to keep secure. That way, those files are easier to back up, copy or transfer. As you've now invested lots of mental energy in understanding how GnuPG works, this is probably the easiest way to start with file encryption.

GnuPG can use the same public and private keys to sign or encrypt files as it does email. But equally, there's nothing stopping you creating a separate key pair purely for dealing with files, although you will lose the ability to sign files you may want other people to authenticate.

It's also possible to create sub-keys off your main key, using sub-keys for different purposes or devices without losing the original authenticity. The only special consideration you should make is whether you need to compress your files first. Properly encrypted data should look no different to a random stream of data. But file compression only works when there's duplication, which makes post-encryption compression a waste of time. For that reason, you'll need to gzip or bzip2 your files and folders before encrypting them, or use your favourite GUI tool.

Encrypting the files is easy, especially now we've already created a pair of keys and the *GPA* tool includes a file manager. This simple utility lets you add unencrypted files using a simple requester, and a button in the toolbar lets you use your GnuPG keys to create an encrypted version, saved automatically with the **gpg** extension. It will also decrypt files in the same way. And this being GnuPG, anyone will be able to work with the same files, regardless of the application or utility, if they've got the prerequisite private or public keys. It's a great way of backing up important files, for example, but it does make it more vital that you remember the passphrase to your key. If this is lost, there's no way of decrypting your privately encrypted files. To solve this, ensure your encrypted key is saved in several places. Due to the strength of the encryption some people even leave these in plain sight, such as synchronised to their public GitHub accounts, but cloud-based email is another good option.


random.txt — Okteta
File Edit View Windows Bookmarks Tools Settings Help
New Open Save Save As Undo Redo Cut Copy Paste Find Find Next
raydon_s_Media_and_File_Server_Bundle_for_the_Foxsat_HDR_release_4.1.3.rar.gpg
random.txt

There should be no patterns or discernible information in encrypted data. It should appear as random as the output from **/dev/urandom**, as shown here.

Encrypt your home folder

Taking file encryption a step further, you might want to consider encrypting your home folder, decrypting it live as you log in. This is what Ubuntu does when you enable the aforementioned installation-time encryption.

Its obvious advantage over per-file or per-folder encryption is that you don't have to think about what you need to

The configuration files of GnuPG

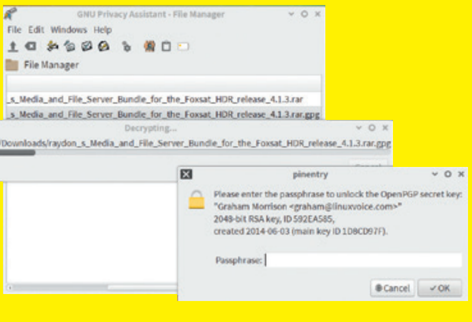
Even with the GUI approach, it still helps to have some understanding of what files and folders are typically stored in **.gnupg** and what part of the encryption process they're responsible for, regardless of whether you use the command line or a GUI tool. In doing so, you'll find using GnuPG easier, and it will be less likely you'll make a mistake (such as publishing your private keys on GitHub).

pubring.gpg This file contains your collection of public keys. These are public keys from people you meet, people who contact you with their key, and keys downloaded from a trusted key server.

secring.gpg Along with **pubring.gpg**, this is the other vital component in a configured GnuPG setup. It contains your secret key. However, from GnuPG version 2.1 onwards, this file becomes deprecated in favour of a new agent that automatically places secret keys into a folder called **private-keys-v1.d** (any files you have will be automatically moved).

crls.d The **crl** in this folder name represents 'Certificate Revocation List'. It's a way of marking keys you have generated as invalid - you might have sent a private key by mistake, or lost the passphrase, for example.

gpg.conf This is standard Linux procedure – the configuration file that tells GnuPG what it needs to know. If you use the command line, you can use it to hold your preferred values – such as which key to use by default, or which keyserver you prefer.


GNU Privacy Assistant - File Manager
File Edit Windows Help
File Manager
_s_Media_and_File_Server_Bundle_for_the_Foxsat_HDR_release_4.1.3.rar
_s_Media_and_File_Server_Bundle_for_the_Foxsat_HDR_release_4.1.3.rar.gpg
Decrypting...
/home/graham/Downloads/raydon_s_Media_and_File_Server_Bundle_for_the_Foxsat_HDR_release_4.1.3.rar.gpg
pinentry
Cancel
OK

Please enter the passphrase to unlock the OpenPGP secret key:
"Graham Morrison <graham@linuxvoice.com>"
2048-bit RSA key, ID 592EA585,
created 2014-06-03 (main key ID 1DBCD97F).

Passphrase:

GnuPG can be used to encrypt and sign your files, using the command line or one of its front-ends.

keep secure: everything is secure. If your laptop is off, your data is inaccessible without your passphrase, which is usually configured to be your login password. However, you still need to make sure temporary files or caches used by your system are either saved to your home folder or deleted on shutdown. You also need to make sure you enable swap partition encryption, as the swap file contains data shuffled in and out of your system memory. We've been using an SSD laptop without a swap partition for a couple of years without issue, which does sidestep this problem.

If your laptop is off, your data is inaccessible without your passphrase, which is usually your login password

The tool we're going to use is the aforementioned eCryptfs. Ubuntu configuration is easy, thanks to Canonical's developers. But those easy scripts for creating and managing eCryptfs have been migrated to most other distributions too, so it's almost as easy on Fedora and even Arch. Just search for and install the **ecryptfs-utils** package, which should also ask for **rsync** and **lsof** to help with locating a user's open files. eCryptfs itself is part of the kernel, and the module needs to be loaded before you can start (**sudo modprobe ecryptfs**), although this will be done automatically after you've created an encrypted folder.

It's also important to note that eCryptfs sits on top of the filesystem – it's not formatting your data at the block level. It's mounted using FUSE after your system starts, making it much easier to manage and copy encrypted data. Before mounting, the encrypted data can be found in a user's **.Private** folder, but after mounting, your home folder will look just as it should.

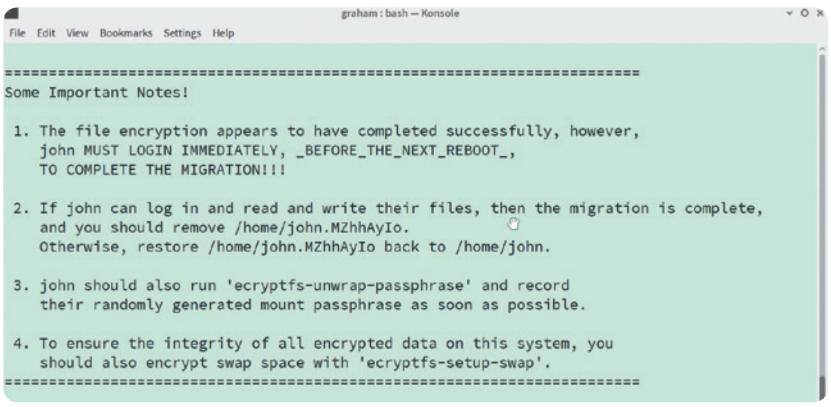
John the Revelator

Choose a user whose home folder you want to encrypt (we're going with 'john'), and without the user being logged in, check to make sure they've no processes running. You can do this by typing **ps -u john**. John should also make sure their password is tough yet memorable, as this will be used as the passphrase to encrypt their data. Depending on the size of that user's home folder – you can check by typing **du -h /home/john** – it may be quicker and easier to move out the majority of that user's data. The first stage of encryption will be far quicker, and the user can then move the data back as they need it. There's no point encrypting a download folder full of ISOs, for example. You can now instantiate the migration process with a single command:

```
sudo ecryptfs-migrate-home -u john
```

You'll be asked for John's login password before the encryption process kicks off. When it's finished, text output will inform you of the next step, and if you check, you'll see the contents of John's home folder has now disappeared. Don't worry, though, it's hidden in the **/home** folder for now, just in case something has gone wrong. John should log in and type **ecryptfs-mount-private** to mount the decrypted folder and check it contains the expected files. Follow this with **ecryptfs-unwrap-passphrase** to reveal the master key in the encryption. Keep a note of this, as it will be required if you need to access the data outside of the login account environment, perhaps using GnuPG to keep it safe and stored somewhere else.

Finally, when you're happy that the configuration is working, remove the hidden /home folder with the unencrypted versions and reboot.



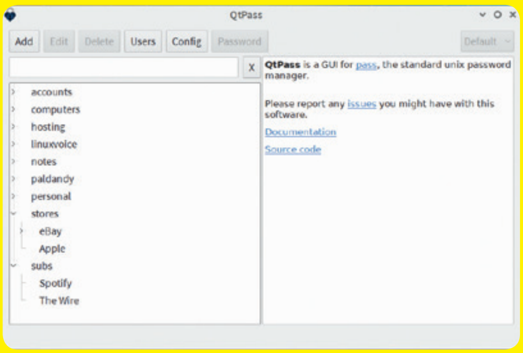
```
========================================================================
Some Important Notes!

 1. The file encryption appears to have completed successfully, however,
    john MUST LOGIN IMMEDIATELY, _BEFORE_THE_NEXT_REBOOT_,
    TO COMPLETE THE MIGRATION!!!

 2. If john can log in and read and write their files, then the migration is complete,
    and you should remove /home/john.MZhhAyIo.
    Otherwise, restore /home/john.MZhhAyIo back to /home/john.

 3. john should also run 'ecryptfs-unwrap-passphrase' and record
    their randomly generated mount passphrase as soon as possible.

 4. To ensure the integrity of all encrypted data on this system, you
    should also encrypt swap space with 'ecryptfs-setup-swap'.
========================================================================
```

Thanks to some excellent eCryptfs utility scripts, encrypting any folder is an easy task.

Encrypting passwords

There are many ways of storing important information like your various password and passphrase, but the best we've found is a small tool called simply *pass*. The great advantage with *pass* is that it uses your filesystem to store and organise files that contain the data. You could create a folder called **Shopping**, for instance, and use this to hold password to your Amazon, eBay and Google accounts.

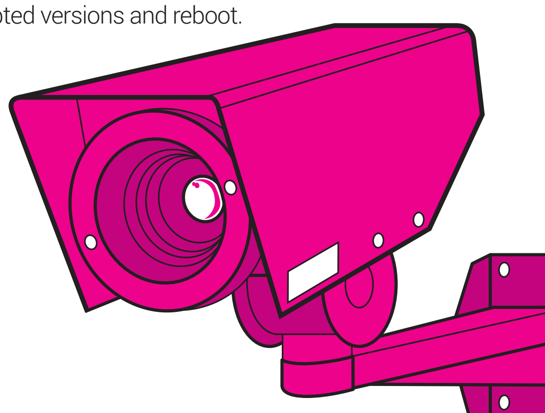
As the passwords themselves are encrypted using the same GnuPG utilities we've already covered, you can access their contents even when you're not using the *pass* command. It works particularly well when you use a remote synchronisation service, such as the one offered by OwnCloud, to automatically copy the contents of your password folders to an online backup or remotely shared storage. We covered the specifics of *pass* in issue 19's FOSSPicks, but you can skip the command line entirely by using one of the many GUIs created to handle the complexity automatically. Our favourite for any desktop is *QtPass*, and we use an app called *Password Store* on Android to access the same shared passwords.
www.passwordstore.org



QtPass is a GUI for pass, the standard unix password manager.

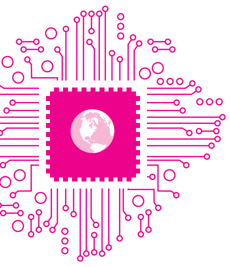
Please report any issues you might have with this software.

Pass transparently uses the power of GnuPG and your filesystem to intelligently store passwords and any other in data you want to keep secret.



NETWORKING

Secure your connection with the magic of OpenVPN.



Your home network is mostly a trusted environment. You can plug devices into your network and not worry too much about the data as it travels between your NAS, your light switches and your Raspberry Pi.

Your router to the internet is your firewall, and most things behind it are safe. But as soon as your data passes through that firewall, as soon as you try to turn up the heating from your office before heading home, that data is subjected to the wilds of the internet, a place where no packet is safe. And that's before we've even factored in the nightmare of open Wi-Fi access points.

The solution is to tunnel your data from your machine to your remote trusted network. This is what a VPN does, and it's what SSH does too, to a lesser extent. The tunnel is protected from the wilds of the internet by encryption, with keys at either end of the tunnel encrypting and decrypting the data for transmission and upon arrival. Anything looking at the data in-between will see a stream of random bits and bytes. They won't even be able to tell which protocols you're using or how you're using them.

VPNs are becoming incredibly popular because they're the most comprehensive solution when you need to leapfrog an insecure network or access machines behind a firewall. When you're connected to a VPN, your local connection behaves exactly as if it were physically relocated to the remote location, accessing those boxes and services without further configuration. If you access your BrewPi at home with the IP address 192.168.1.177, you'll use the same IP address if you use a VPN to access your home network from a remote location.

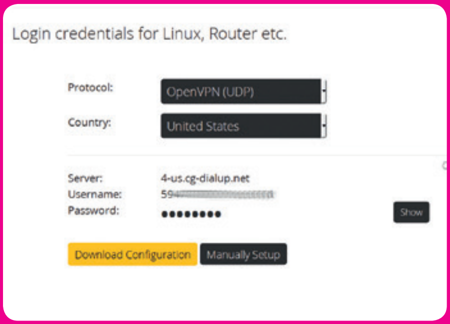
Connecting to a VPN

It's highly likely you've already used a VPN for work, as they enable remote workers to access the same facilities as when they're in the office, only securely from a remote location. But over the last couple of years there's been a proliferation of low-cost private VPN providers offering to tunnel your connection to some remote geographical location while also vetting your data for security and privacy, as well as ad blocking. These are perfect for side-stepping online restrictions and local security issues.

The vast majority of VPNs support OpenVPN, an open source VPN that uses OpenSSL for encryption. OpenVPN is almost ubiquitous on Linux systems, which means that configuration should be straightforward whether you're connecting via the desktop, a Cyanogenmod phone or a Linux router. We've found the most reliable method for configuring and enabling OpenVPN is via the **openvpn** command line tool. With this installed and the **ovpn** file bundle downloaded from your VPN provider and unzipped, simply configure an OpenVPN connection by typing **sudo openvpn --config 'path-to-file.ovpn'**. Within a few moments your network connections will become re-routed through the VPN. The best way of proving this from the command line is to try and access a few sites, or use a website to check your IP and geographical location. If the VPN is working, your location will appear to be where the VPN exit is located, not the location of your local network provider.

> The vast majority of VPNs support OpenVPN, an open source VPN that uses SSL for encryption

STEP BY STEP: OPENVPN VIA A NETWORKMANAGER GUI

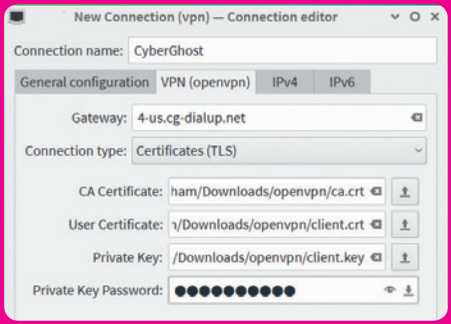


Login credentials for Linux, Router etc.

Protocol: OpenVPN (UDP)
Country: United States

Server: 4-us.cg-dialup.net
Username: 59[redacted]
Password: ●●●●●●●●●

1 Connection editor
Most distros have now moved to *Systemd*'s network manager for their networking duties. Both Gnome's and KDE's GUI for this will let you add OpenVPN connections and import a config file. You'll first need to download the certificate and **ovpn** file bundle available from most OpenVPN servers.

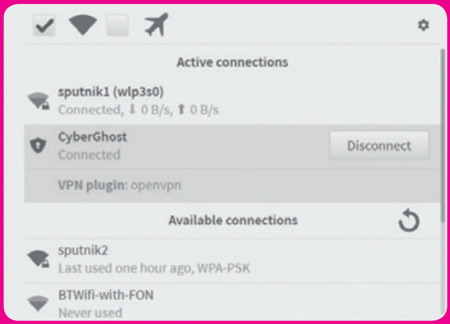


New Connection (vpn) — Connection editor

Connection name: CyberGhost

Gateway: 4-us.cg-dialup.net
Connection type: Certificates (TLS)
CA Certificate: ham/Downloads/openvpn/ca.crt
User Certificate: ı/Downloads/openvpn/client.crt
Private Key: /Downloads/openvpn/client.key
Private Key Password: ●●●●●●●●●●●

2 Configuration
Taking KDE as an example, open the connection editor and use the File menu to select Import VPN. Navigate to the location of your **ovpn** file – unzipping the download archive if necessary, and let the connection manager import your settings. We needed to to add a username and password.



Active connections

sputnik1 (wlp3s0)
Connected, ⬇ 0 B/s, ⬆ 0 B/s

CyberGhost
Connected

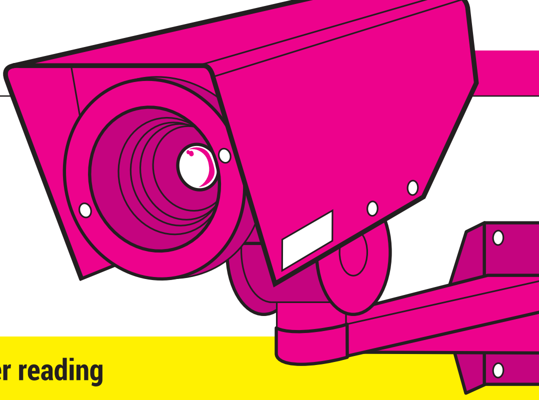
VPN plugin: openvpn

Available connections

sputnik2
Last used one hour ago, WPA-PSK

BTWifi-with-FON
Never used

3 Test VPN
After connecting to your network, you can activate the OpenVPN connection by clicking on the configuration in Network Manager. When this is running, you should see a padlock added to your connection and it should be obvious your data is going through this tunnel.



OPENSSH

Talk to your Linux machines across the wild, wild internet.

We couldn't write about encryption without looking into the open implementation of the SSH protocol, which is also a great place to end our encryption coverage for this issue. This wonderful offshoot of the OpenBSD project is one of the most used and useful tools of the Linux stack, enabling any two Linux boxes, or OpenBSD boxes, to talk securely to one another across the wilds of the internet.

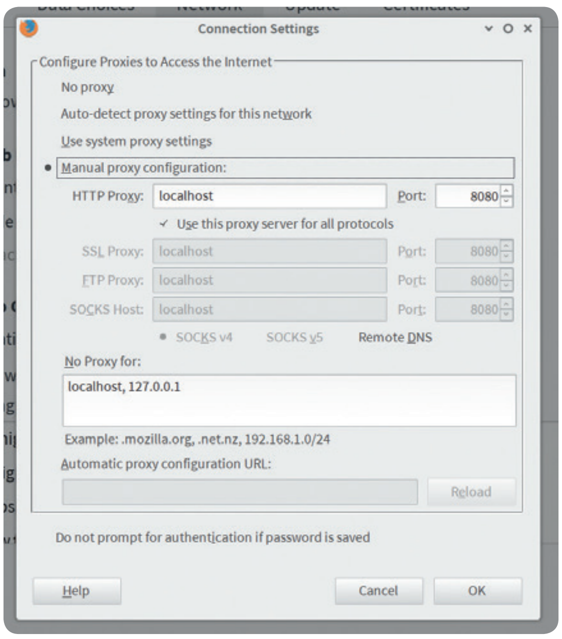
Because SSH uses very little bandwidth to give you a remote terminal, you can perform almost any task that you can locally, even on a desktop, using nothing but a mobile phone and a local GPRS network. We doubt you need reminding of how it works, but things don't get much simpler than typing **ssh** followed by the IP address of your server. If you've got a user account with the same username, you'll only need to enter your password to get access.

Passwordless login

A much quoted improvement you should make is to disable password logins like these and replace them with the magic of a public/private key pair that not only automates SSH login, but makes your server more secure, as you can't get access simply by guessing a password. To do this, on your client machine (not the server), type **ssh-keygen -t rsa** to generate the keypair. Enter a passphrase if you want the added security of requiring both the key and a password. By default, it will save both keys into your **.ssh** folder. You can automatically copy the public key to your server with the following command: **ssh-copy-id -i .ssh/id_rda.pub user@server**.

You can now connect to your remote server without entering a password. If not, make sure the permissions are set correctly (**chmod 770 .ssh**). Finally, if everything is working, disable password logins on the server by commenting out the **PasswordAuthentication** line in the server's **/etc/ssh/sshd_config** configuration file – and don't lose your local SSH keys!

One feature of OpenSSH that isn't often mentioned is that it's also capable of acting as a cheap VPN, tunnelling general requests through the same SSH session you're using for terminal access. It's not a fully fledged solution in the same way as OpenVPN – it uses TCP, for a start, and OpenVPN should ideally be configured to use UDP, as it's faster and better suited to passing through NAT connections, but it's ideal for *ad-hoc* connections when you're away from home, or need to jump a geographical restriction to your low-end-box. The command for doing this is **ssd -D 8080 username@server**, setting up a SOCKS proxy on port 8080 of your local machine. You then need to configure either your computer (desktop) or web browser to pass traffic through this. In *Firefox*, open the Advanced settings pane, click on Network and click on the Settings button to the right of Connection. From the window that appears, enable manual proxy configuration and enter 'localhost' and 8080 as the port. This will re-route all web traffic through the SSH tunnel. You can also do this for a single port with the command **ssh -L 8080:localhost:8080 username@server**, which is useful if you want to tunnel a single service from far away to your local machine. We use this feature to access our router configuration page from a firewall we've configured to only allow through SSH.



Connection Settings

Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- ● Manual proxy configuration:

HTTP Proxy: localhost Port: 8080

✓ Use this proxy server for all protocols

SSL Proxy: localhost Port: 8080

FTP Proxy: localhost Port: 8080

SOCKS Host: localhost Port: 8080

● SOCKS v4 SOCKS v5 Remote DNS

No Proxy for:

localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

Do not prompt for authentication if password is saved

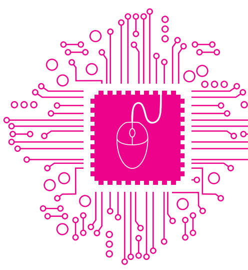
You can easily tunnel web requests through SSH with the -D option to create a SOCKS proxy.

Further reading

Encryption, security and privacy have been an important part of our editorial coverage these last two years, and our old articles are the best place to start if you want to take encryption further (quick plug: all subscribers get access to our entire back issue catalogue).

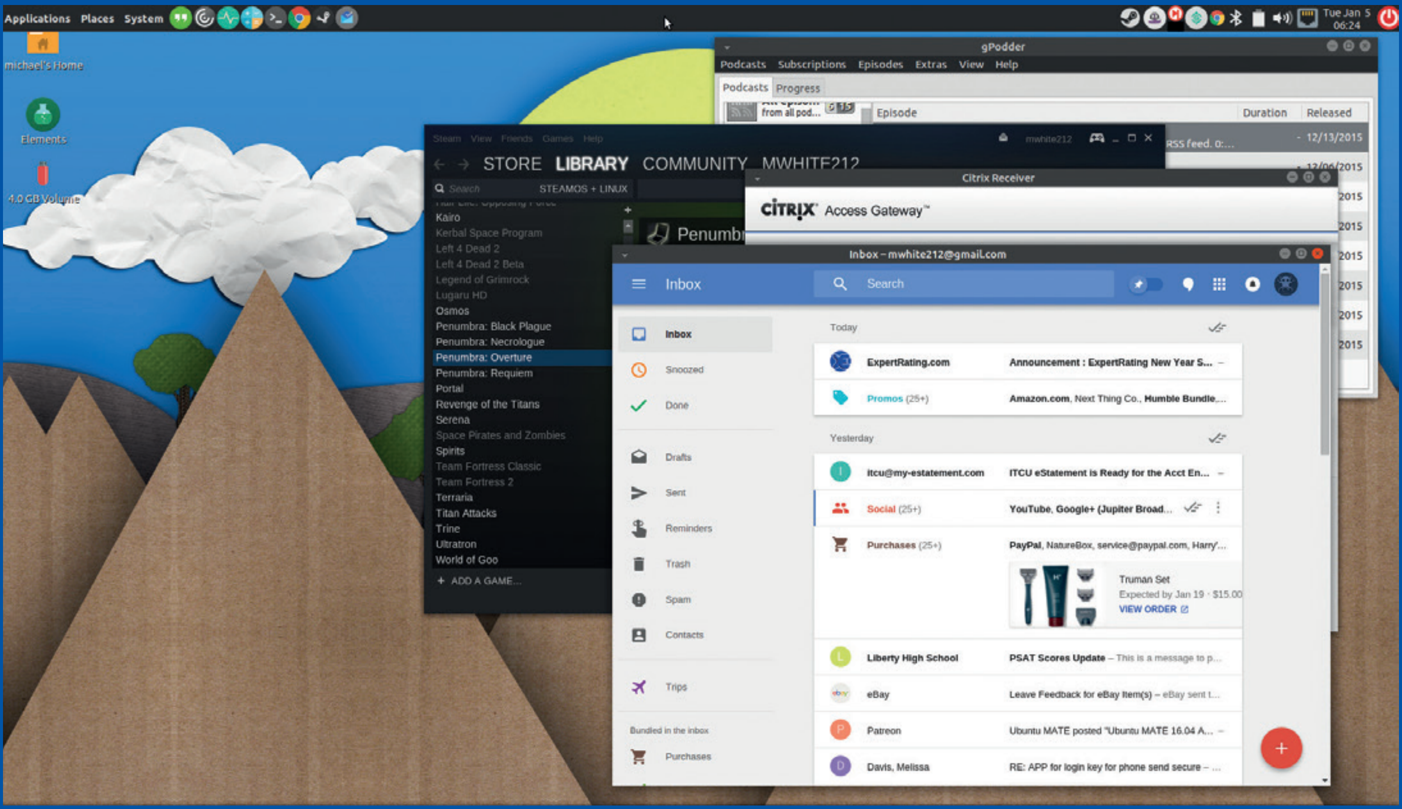
In particular, in issue 15 Jake Margason wrote a brilliant tutorial on hiding encrypted volumes in plain site, so that even the presence of your data becomes ambiguous. We also covered the popular password store, *KeePassX*, in issue 19, which is a great alternative to the humble *pass* tool we mention here. Issue 17 was dedicated to privacy, and Ben's excellent cover feature goes in-depth on how snoopers get access to your data and how how to use tools like *Tor* to keep prying eyes away. There's also a great interview with Cory Doctorow. Finally, as one last tip – keep your distribution up to date. Vulnerabilities will be found even in tools like GnuPG and SSH, and the only solution is to replace those old versions with new ones.

If you want to stay ahead of security and privacy issues, we can think of no better source than Bruce Schneier's blog (https://www.schneier.com).



DESKTOP SHOWCASE

Want to change the way your desktop looks? Searching for a new Linux setup? Here's a selection from Linux Voice readers to inspire you…

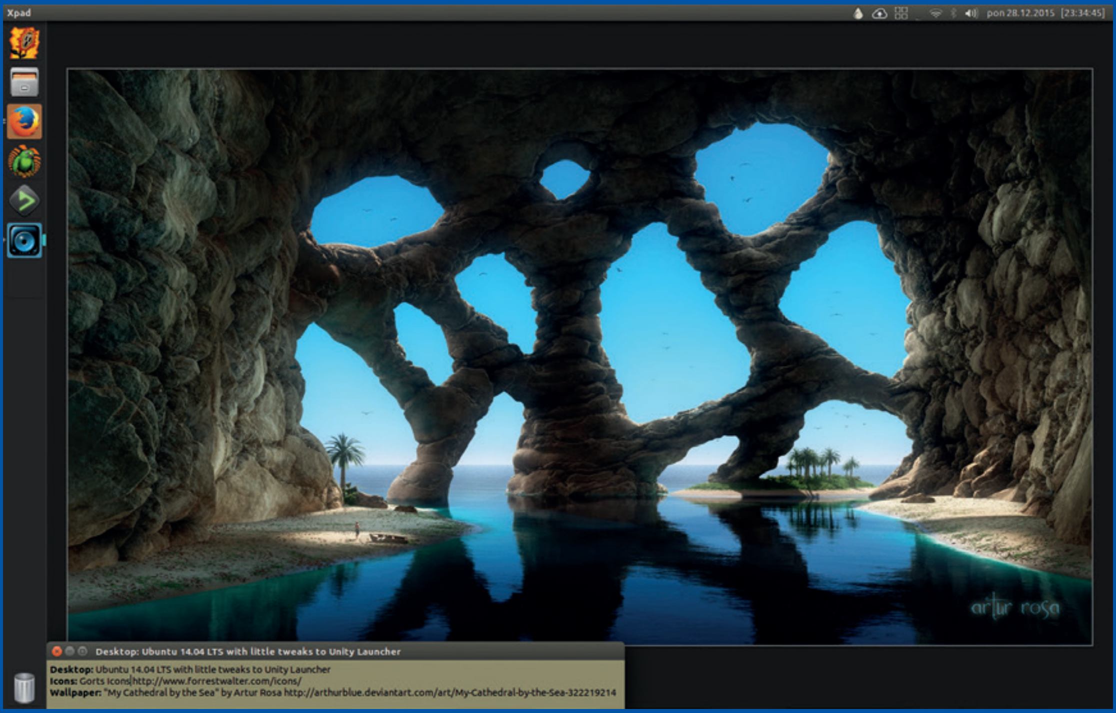


▲

Michael White: Nothing too fancy. Just Ubuntu Mate 15.10 with the Numix Circle icon set and flat window borders. I like it all to look like paper. It pleases me.

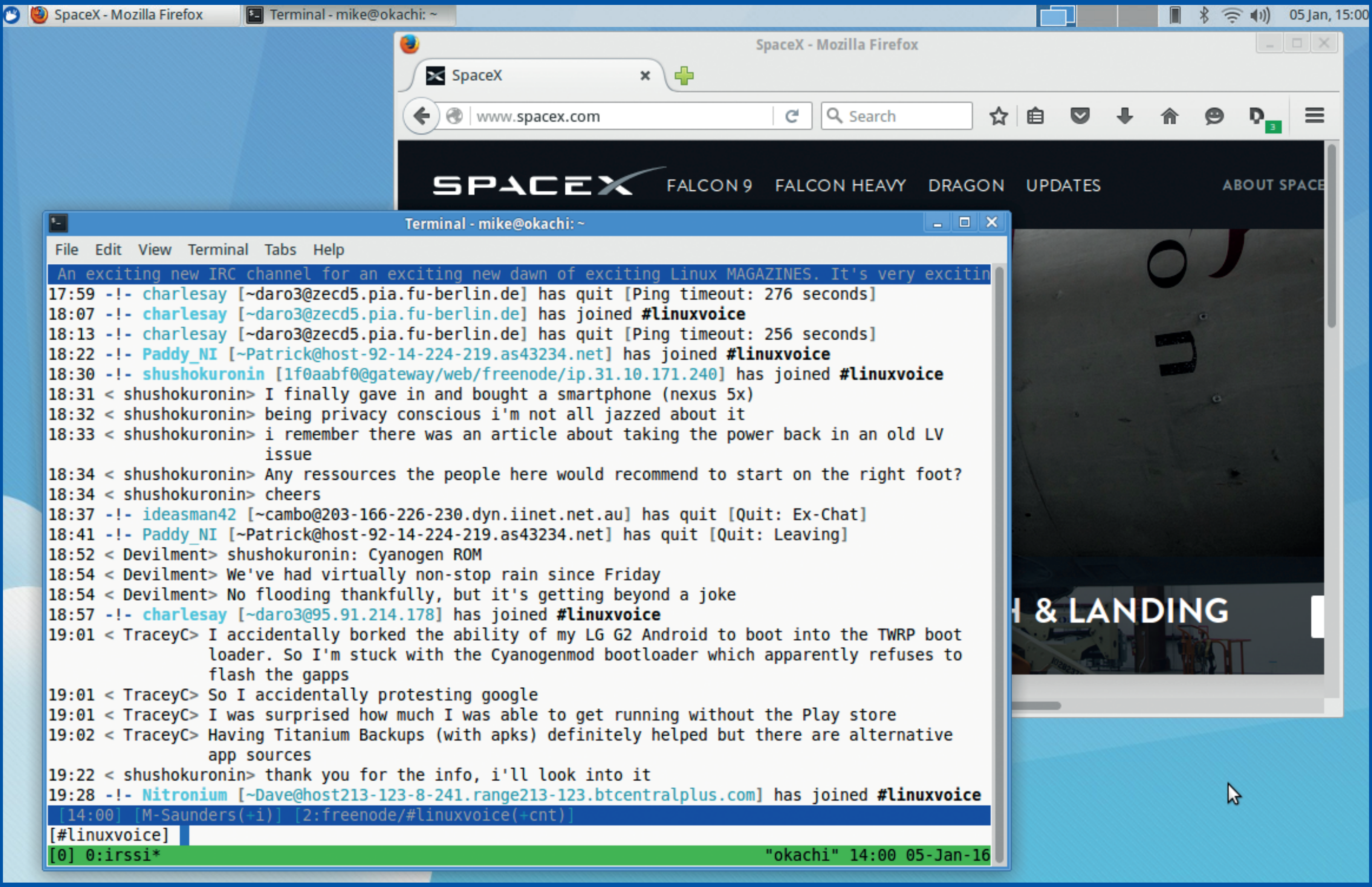
▶

Qkiel: Ubuntu 14.04 LTS with little tweaks to the Unity launcher. The icons are Gorts Icons Vol. 6 (**www.forrestwalter.com/icons**). Wallpaper: "My Cathedral by the Sea" by Artur Rosa (**http://arthurblue.deviantart.com/art/My-Cathedral-by-the-Sea-322219214**).

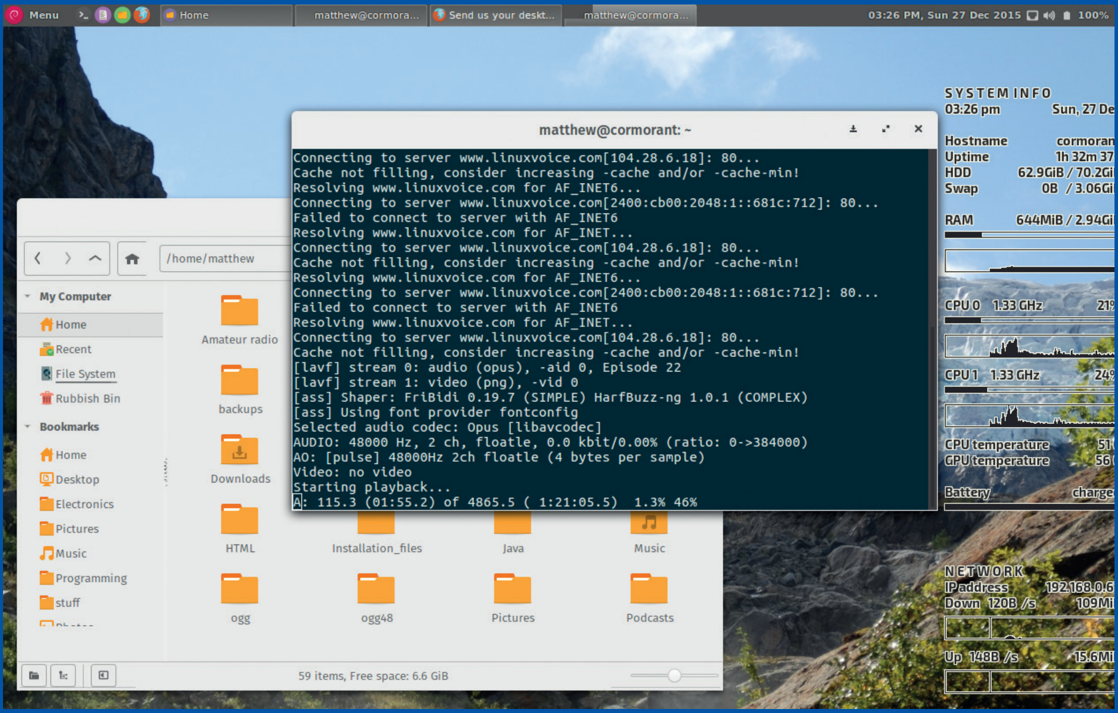


Desktop: Ubuntu 14.04 LTS with little tweaks to Unity Launcher

Desktop: Ubuntu 14.04 LTS with little tweaks to Unity Launcher
Icons: Gorts Icons|http://www.forrestwalter.com/icons/
Wallpaper: "My Cathedral by the Sea" by Artur Rosa http://arthurblue.deviantart.com/art/My-Cathedral-by-the-Sea-322219214



```
 An exciting new IRC channel for an exciting new dawn of exciting Linux MAGAZINES. It's very excitin
17:59 -!- charlesay [~daro3@zecd5.pia.fu-berlin.de] has quit [Ping timeout: 276 seconds]
18:07 -!- charlesay [~daro3@zecd5.pia.fu-berlin.de] has joined #linuxvoice
18:13 -!- charlesay [~daro3@zecd5.pia.fu-berlin.de] has quit [Ping timeout: 256 seconds]
18:22 -!- Paddy_NI [~Patrick@host-92-14-224-219.as43234.net] has joined #linuxvoice
18:30 -!- shushokuronin [1f0aabf0@gateway/web/freenode/ip.31.10.171.240] has joined #linuxvoice
18:31 < shushokuronin> I finally gave in and bought a smartphone (nexus 5x)
18:32 < shushokuronin> being privacy conscious i'm not all jazzed about it
18:33 < shushokuronin> i remember there was an article about taking the power back in an old LV
                       issue
18:34 < shushokuronin> Any ressources the people here would recommend to start on the right foot?
18:34 < shushokuronin> cheers
18:37 -!- ideasman42 [~cambo@203-166-226-230.dyn.iinet.net.au] has quit [Quit: Ex-Chat]
18:41 -!- Paddy_NI [~Patrick@host-92-14-224-219.as43234.net] has quit [Quit: Leaving]
18:52 < Devilment> shushokuronin: Cyanogen ROM
18:54 < Devilment> We've had virtually non-stop rain since Friday
18:54 < Devilment> No flooding thankfully, but it's getting beyond a joke
18:57 -!- charlesay [~daro3@95.91.214.178] has joined #linuxvoice
19:01 < TraceyC> I accidentally borked the ability of my LG G2 Android to boot into the TWRP boot
                 loader. So I'm stuck with the Cyanogenmod bootloader which apparently refuses to
                 flash the gapps
19:01 < TraceyC> So I accidentally protesting google
19:01 < TraceyC> I was surprised how much I was able to get running without the Play store
19:02 < TraceyC> Having Titanium Backups (with apks) definitely helped but there are alternative
                 app sources
19:22 < shushokuronin> thank you for the info, i'll look into it
19:28 -!- Nitronium [~Dave@host213-123-8-241.range213-123.btcentralplus.com] has joined #linuxvoice
 [14:00] [M-Saunders(+i)] [2:freenode/#linuxvoice(+cnt)]
[#linuxvoice]
[0] 0:irssi*                                                              "okachi" 14:00 05-Jan-16
```



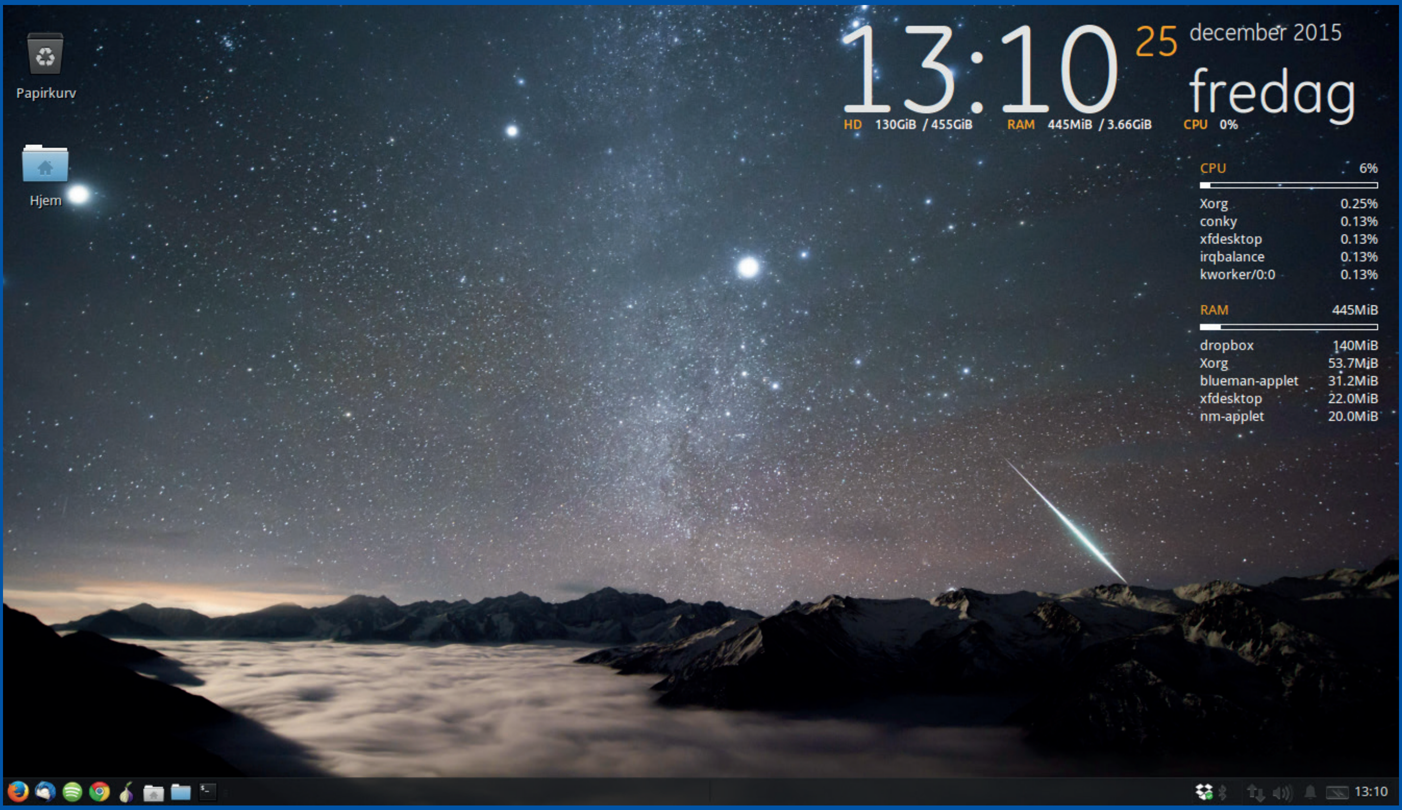
```
Connecting to server www.linuxvoice.com[104.28.6.18]: 80...
Cache not filling, consider increasing -cache and/or -cache-min!
Resolving www.linuxvoice.com for AF_INET6...
Connecting to server www.linuxvoice.com[2400:cb00:2048:1::681c:712]: 80...
Failed to connect to server with AF_INET6
Resolving www.linuxvoice.com for AF_INET...
Connecting to server www.linuxvoice.com[104.28.6.18]: 80...
Cache not filling, consider increasing -cache and/or -cache-min!
Resolving www.linuxvoice.com for AF_INET6...
Connecting to server www.linuxvoice.com[2400:cb00:2048:1::681c:712]: 80...
Failed to connect to server with AF_INET6
Resolving www.linuxvoice.com for AF_INET...
Connecting to server www.linuxvoice.com[104.28.6.18]: 80...
Cache not filling, consider increasing -cache and/or -cache-min!
[lavf] stream 0: audio (opus), -aid 0, Episode 22
[lavf] stream 1: video (png), -vid 0
[ass] Shaper: FriBidi 0.19.7 (SIMPLE) HarfBuzz-ng 1.0.1 (COMPLEX)
[ass] Using font provider fontconfig
Selected audio codec: Opus [libavcodec]
AUDIO: 48000 Hz, 2 ch, floatle, 0.0 kbit/0.00% (ratio: 0->384000)
AO: [pulse] 48000Hz 2ch floatle (4 bytes per sample)
Video: no video
Starting playback...
A: 115.3 (01:55.2) of 4865.5 ( 1:21:05.5)  1.3% 46%
```

▲

Mike Saunders: Xubuntu 15.10 on my Libreboot-modified ThinkPad X200. I've to made the theme lighter and brighter, and I'm checking out #linuxvoice on **chat. freenode.net**, waiting for the next SpaceX launch...

◀

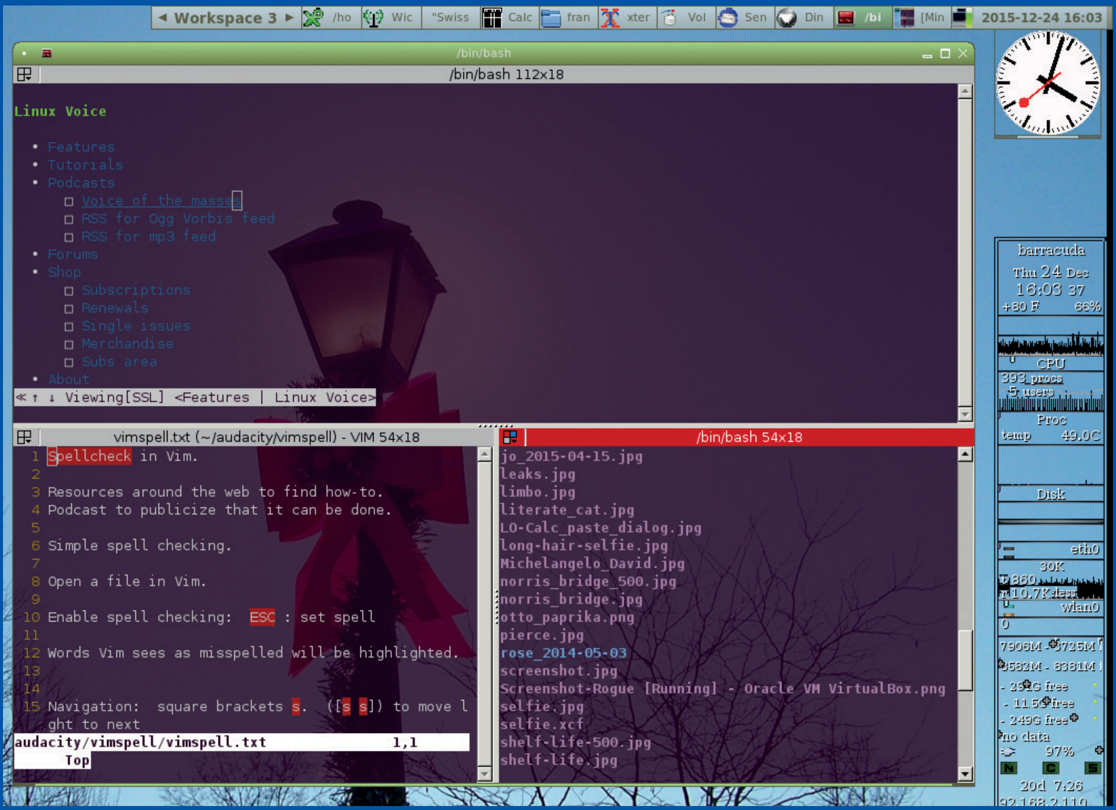
Matt B: I'm currently running Cinnamon on Debian testing, along with the Orchis *GTK* theme and Numix circle icons, and Conky in the background.



```
13:10  25 december 2015
          fredag
HD  130GiB / 455GiB   RAM  445MiB / 3.66GiB   CPU  0%

CPU                 6%
Xorg             0.25%
conky            0.13%
xfdesktop        0.13%
irqbalance       0.13%
kworker/0:0      0.13%

RAM             445MiB
dropbox         140MiB
Xorg            53.7MiB
blueman-applet  31.2MiB
xfdesktop       22.0MiB
nm-applet       20.0MiB
```

▲

Sandersson: Xubuntu with panel in the button. Wallpaper from WallBase. Conky/Gotham and TeejeeTech Process Panel. GreyBird Style. Elementary icons.



```
Linux Voice

  • Features
  • Tutorials
  • Podcasts
      ◻ Voice of the masses
      ◻ RSS for Ogg Vorbis feed
      ◻ RSS for mp3 feed
  • Forums
  • Shop
      ◻ Subscriptions
      ◻ Renewals
      ◻ Single issues
      ◻ Merchandise
      ◻ Subs area
  • About
« ↑ ↓ Viewing[SSL] <Features | Linux Voice>
```

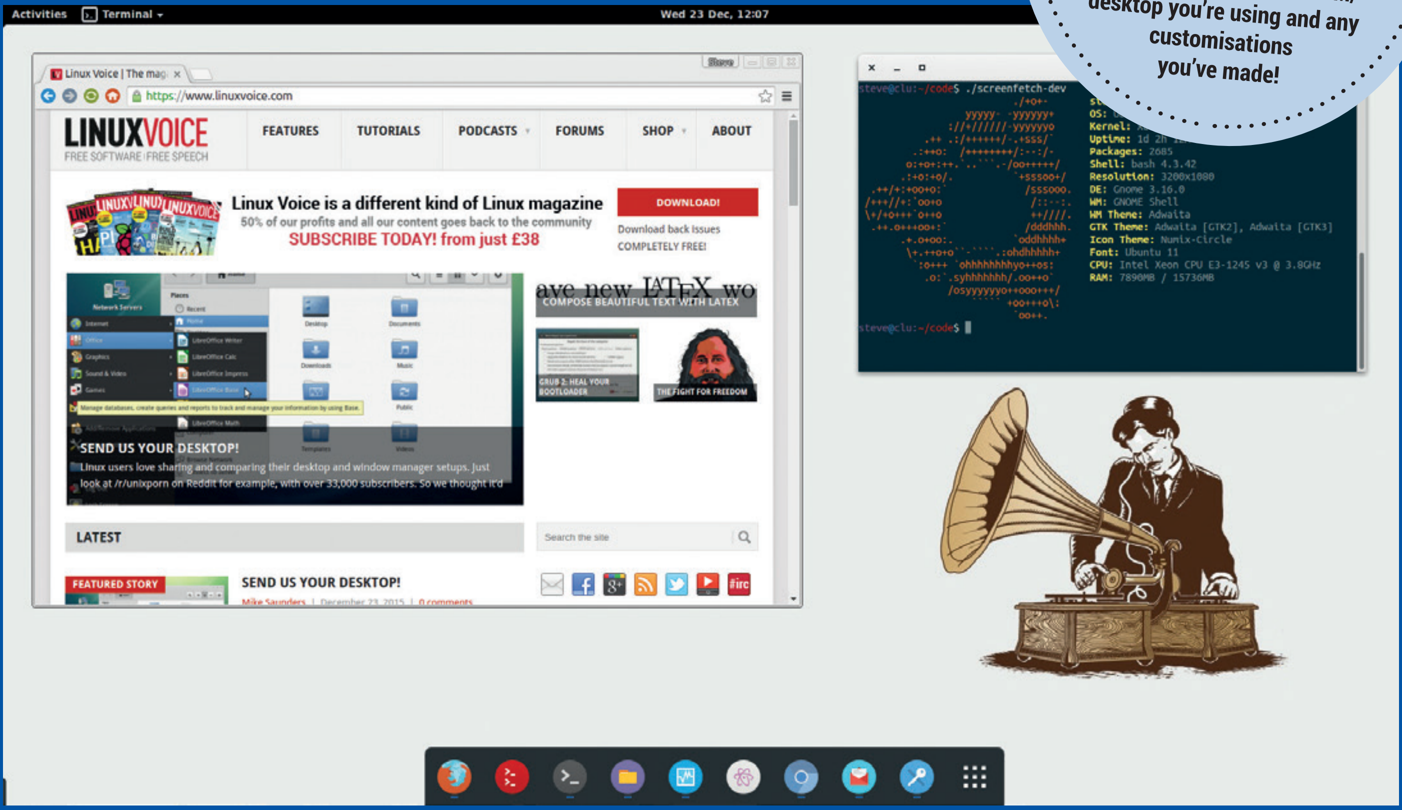
```
Spellcheck in Vim.

Resources around the web to find how-to.
Podcast to publicize that it can be done.

Simple spell checking.

Open a file in Vim.

Enable spell checking:  ESC : set spell

Words Vim sees as misspelled will be highlighted.


Navigation:  square brackets s.  ([s s]) to move l
ght to next
audacity/vimspell/vimspell.txt                1,1
      Top
```

```
jo_2015-04-15.jpg
leaks.jpg
limbo.jpg
literate_cat.jpg
LO-Calc_paste_dialog.jpg
long-hair-selfie.jpg
Michelangelo_David.jpg
norris_bridge_500.jpg
norris_bridge.jpg
otto_paprika.png
pierce.jpg
rose_2014-05-03
screenshot.jpg
Screenshot-Rogue [Running] - Oracle VM VirtualBox.png
selfie.jpg
selfie.xcf
shelf-life-500.jpg
shelf-life.jpg
```

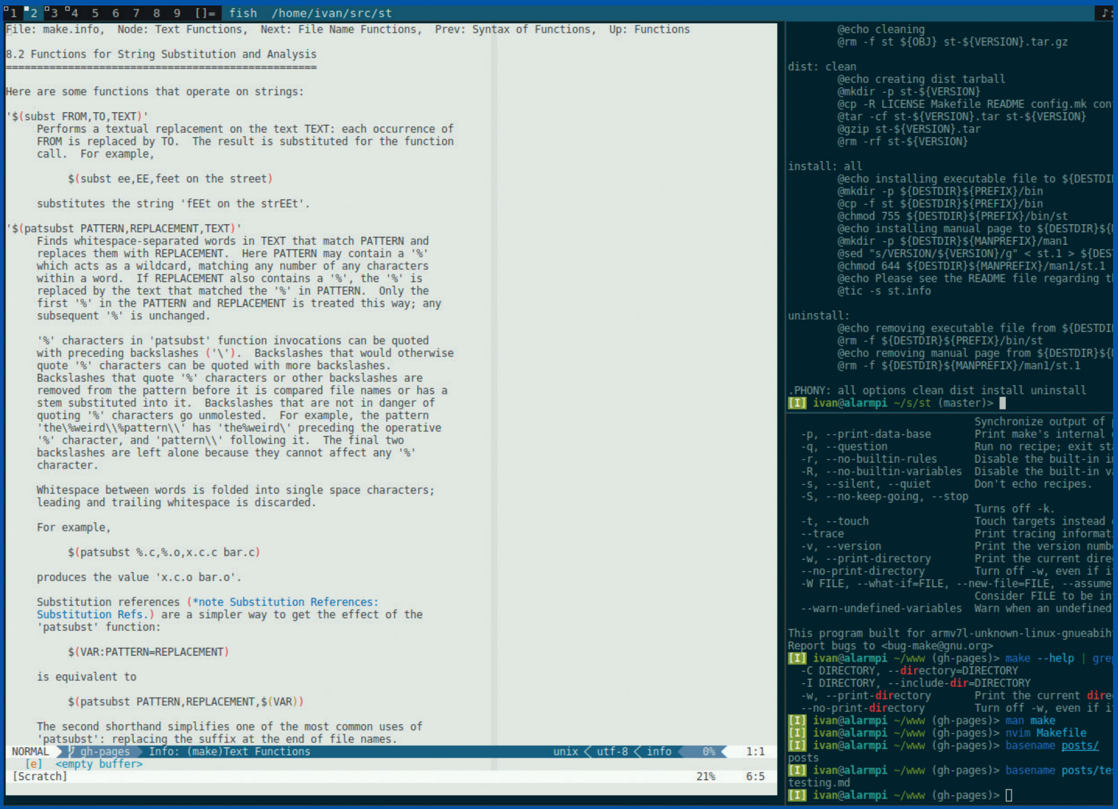
▶

Frank Bell: Mint 17.1 with Fluxbox, Terminator, and GKrellM.

Want your desktop or window manager featured in the magazine? Take a full-screen screenshot in PNG format and mail it to mike@linuxvoice.com. Add some words (max 50) about your setup, such as the WM/desktop you're using and any customisations you've made!



Steve Newbury: Ubuntu 15.10 with Gnome 3.16 – added Numix-Circle Icons, Dash-To-Dock and Dropdown terminal extensions. Solarized Dark terminal theme in Gnome terminal.



```
¹1 ²2 ³3 ⁴4  5  6  7  8  9  []=  fish  /home/ivan/src/st
File: make.info,  Node: Text Functions,  Next: File Name Functions,  Prev: Syntax of Functions,  Up: Functions

8.2 Functions for String Substitution and Analysis
==================================================

Here are some functions that operate on strings:

'$(subst FROM,TO,TEXT)'
     Performs a textual replacement on the text TEXT: each occurrence of
     FROM is replaced by TO.  The result is substituted for the function
     call.  For example,

          $(subst ee,EE,feet on the street)

     substitutes the string 'fEEt on the strEEt'.

'$(patsubst PATTERN,REPLACEMENT,TEXT)'
     Finds whitespace-separated words in TEXT that match PATTERN and
     replaces them with REPLACEMENT.  Here PATTERN may contain a '%'
     which acts as a wildcard, matching any number of any characters
     within a word.  If REPLACEMENT also contains a '%', the '%' is
     replaced by the text that matched the '%' in PATTERN.  Only the
     first '%' in the PATTERN and REPLACEMENT is treated this way; any
     subsequent '%' is unchanged.

     '%' characters in 'patsubst' function invocations can be quoted
     with preceding backslashes ('\').  Backslashes that would otherwise
     quote '%' characters can be quoted with more backslashes.
     Backslashes that quote '%' characters or other backslashes are
     removed from the pattern before it is compared file names or has a
     stem substituted into it.  Backslashes that are not in danger of
     quoting '%' characters go unmolested.  For example, the pattern
     'the\%weird\\%pattern\\' has 'the%weird\' preceding the operative
     '%' character, and 'pattern\\' following it.  The final two
     backslashes are left alone because they cannot affect any '%'
     character.

     Whitespace between words is folded into single space characters;
     leading and trailing whitespace is discarded.

     For example,

          $(patsubst %.c,%.o,x.c.c bar.c)

     produces the value 'x.c.o bar.o'.

     Substitution references (*note Substitution References:
     Substitution Refs.) are a simpler way to get the effect of the
     'patsubst' function:

          $(VAR:PATTERN=REPLACEMENT)

     is equivalent to

          $(patsubst PATTERN,REPLACEMENT,$(VAR))

     The second shorthand simplifies one of the most common uses of
     'patsubst': replacing the suffix at the end of file names.
NORMAL  gh-pages  Info: (make)Text Functions                        unix < utf-8 < info   0%    1:1
  [e]  <empty buffer>
 [Scratch]                                                                                21%    6:5
```

```
        @echo cleaning
        @rm -f st ${OBJ} st-${VERSION}.tar.gz

dist: clean
        @echo creating dist tarball
        @mkdir -p st-${VERSION}
        @cp -R LICENSE Makefile README config.mk con
        @tar -cf st-${VERSION}.tar st-${VERSION}
        @gzip st-${VERSION}.tar
        @rm -rf st-${VERSION}

install: all
        @echo installing executable file to ${DESTDI
        @mkdir -p ${DESTDIR}${PREFIX}/bin
        @cp -f st ${DESTDIR}${PREFIX}/bin
        @chmod 755 ${DESTDIR}${PREFIX}/bin/st
        @echo installing manual page to ${DESTDIR}${
        @mkdir -p ${DESTDIR}${MANPREFIX}/man1
        @sed "s/VERSION/${VERSION}/g" < st.1 > ${DES
        @chmod 644 ${DESTDIR}${MANPREFIX}/man1/st.1
        @echo Please see the README file regarding t
        @tic -s st.info

uninstall:
        @echo removing executable file from ${DESTDI
        @rm -f ${DESTDIR}${PREFIX}/bin/st
        @echo removing manual page from ${DESTDIR}${
        @rm -f ${DESTDIR}${MANPREFIX}/man1/st.1

.PHONY: all options clean dist install uninstall
[I] ivan@alarmpi ~/s/st (master)> 
```

```
                               Synchronize output of
  -p, --print-data-base        Print make's internal
  -q, --question               Run no recipe; exit st
  -r, --no-builtin-rules       Disable the built-in i
  -R, --no-builtin-variables   Disable the built-in v
  -s, --silent, --quiet        Don't echo recipes.
  -S, --no-keep-going, --stop
                               Turns off -k.
  -t, --touch                  Touch targets instead
  --trace                      Print tracing informat
  -v, --version                Print the version numb
  -w, --print-directory        Print the current dire
  --no-print-directory         Turn off -w, even if i
  -W FILE, --what-if=FILE, --new-file=FILE, --assume
                               Consider FILE to be in
  --warn-undefined-variables   Warn when an undefined

This program built for armv7l-unknown-linux-gnueabih
Report bugs to <bug-make@gnu.org>
[I] ivan@alarmpi ~/www (gh-pages)> make --help | gre
  -C DIRECTORY, --directory=DIRECTORY
  -I DIRECTORY, --include-dir=DIRECTORY
  -w, --print-directory        Print the current dire
  --no-print-directory         Turn off -w, even if i
[I] ivan@alarmpi ~/www (gh-pages)> man make
[I] ivan@alarmpi ~/www (gh-pages)> nvim Makefile
[I] ivan@alarmpi ~/www (gh-pages)> basename posts/
posts
[I] ivan@alarmpi ~/www (gh-pages)> basename posts/te
testing.md
[I] ivan@alarmpi ~/www (gh-pages)> 
```

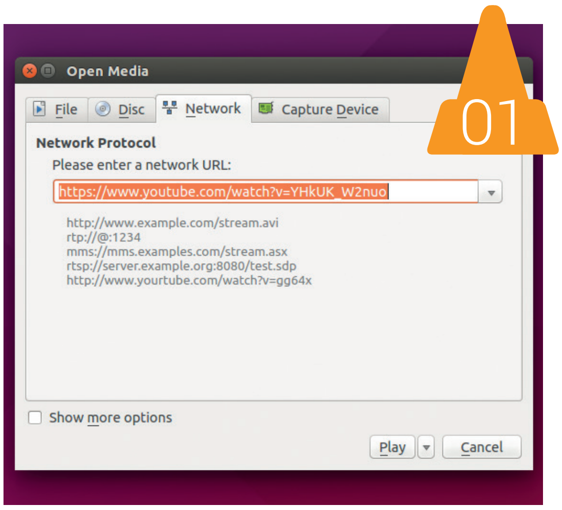
Pickfire: DWM tiling window manager (**http://dwm.suckless.org**), doing some work on Makefiles.

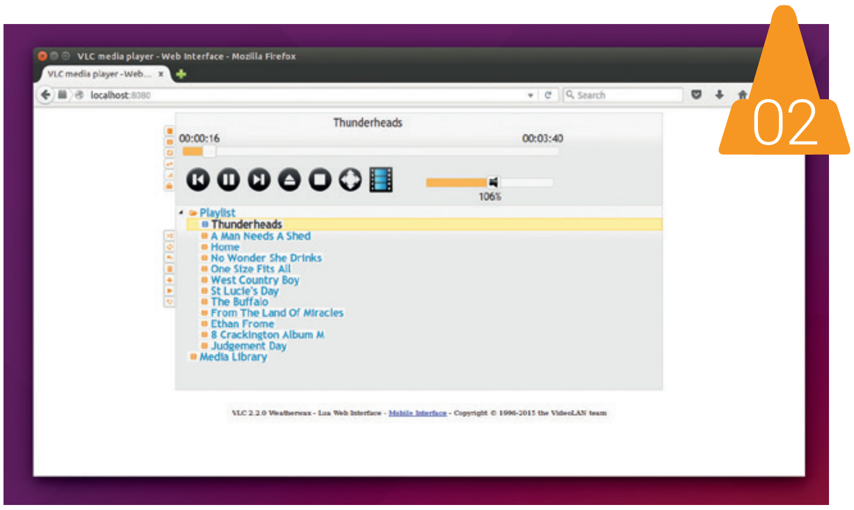
SECRETS OF VLC

Master your media playback.

VLC is among the most popular open source tools on any platform. It works even with files that many other video players struggle to read, and has a simple, clean interface that's easy to use. Below this simple interface, however, there lurk many more powerful features, but they're not always easy to find.

Fear not! For brave Sir Ben has trawled the graphical and command line interfaces of this incredibly useful media player to find the eight most useful of these hidden features covering everything from new media sources to easy control when away from your desk. Never before has watching *Monty Python* been easier!


01
Open Media
File
Disc
Network
Capture Device
Network Protocol
Please enter a network URL:
https://www.youtube.com/watch?v=YHkUK_W2nuo
http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x
Show more options
Play
Cancel



01 Play from YouTube

The YouTube web page works well for short videos, but if you're settling down for something longer, you might want a little more control, perhaps over the audio and video (see tip 7) or over playback control (see tip 2). In these cases, you can use *VLC* as the playback device for YouTube videos. First, you need to use your browser to navigate to the video you want to watch, then open *VLC* and go to Media > Open Network Stream. Paste in the YouTube link, and it will play in the application.

02 Browser remote

When watching videos, it's often not convenient to use your mouse or keyboard to control playback, but fortunately *VLC* includes a web server that can host a simple web page with controls for the video. The easiest way to start this is by launching *VLC* from the command line with the following:

```
vlc -I http --http-password pass -I Qt
```

This uses **pass** as the password; if you're on a public network, you may want to use something more secure. Open **http://<ip-address>:8080** to use the controls.

03 Convert formats

Video players are fickle beasts, and just because you use *VLC*, which plays just about everything under the sun, that doesn't mean you can ignore the problems of others. Fortunately, *VLC* can convert between formats to make movies more friendly for other devices. Go to File > Convert / Save, then select the file to be converted and then press, Convert/ Save. In the new window, select the output options and press start Start. This method can also be used to save network videos such as those from YouTube.

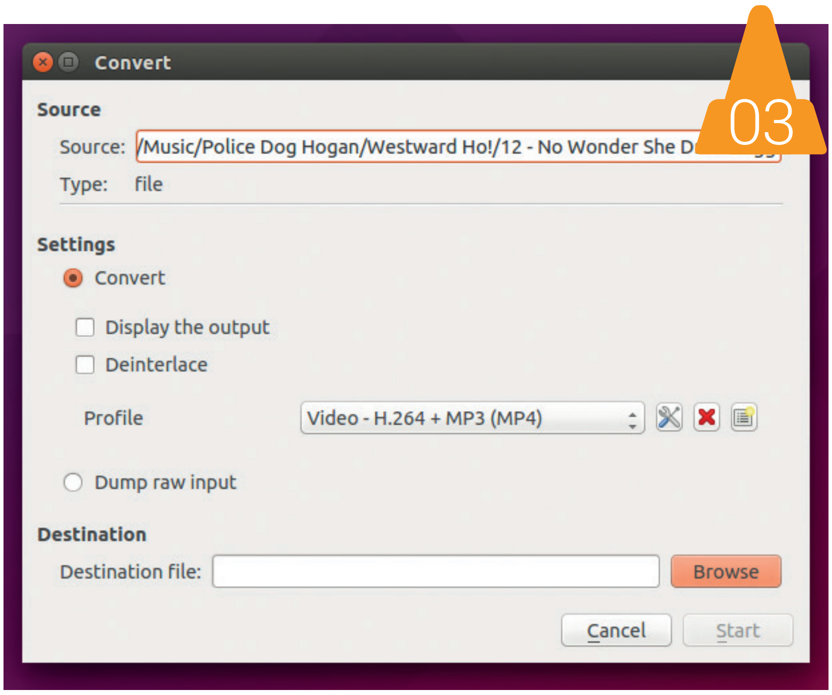
> Sometimes videos may need cropping, noise reduction or adjusting the sync between video and audio – VLC can do all this on the fly

04 Capture video

As well as playing video, *VLC* can help you shoot one. Go to File > Open Capture Device, select the capture mode you want (probably either Video Camera or Desktop), and you can start recording. It won't be able to replace proper recording software for serious users, but it's more than capable for simple messing about.

05 Listen to radio and podcasts

VLC is a general media player, and as well as video, it plays audio files. You can play these in the same way you would with videos, or grab them from the internet. If you



Source: /Music/Police Dog Hogan/Westward Ho!/12 - No Wonder She D

Type: file

[✓ Convert]

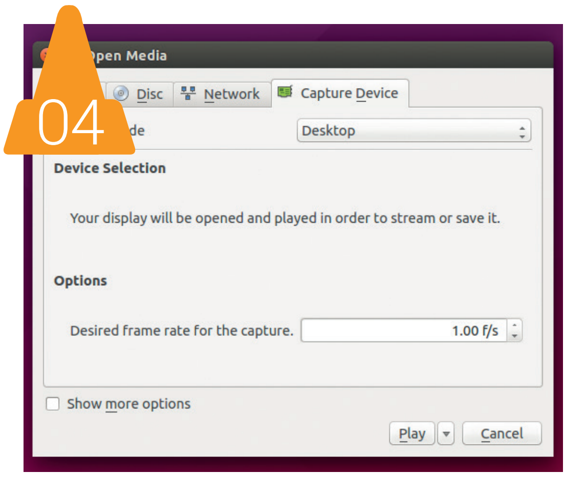
Display the output

Deinterlace

Profile: Video - H.264 + MP3 (MP4)

Dump raw input

Destination file:



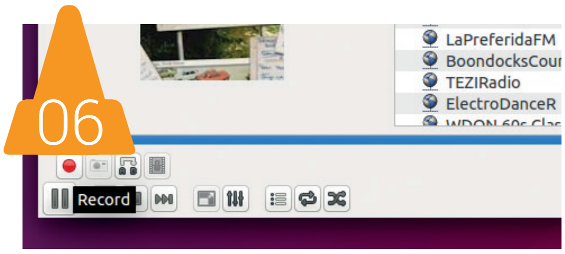
Capture mode: Desktop

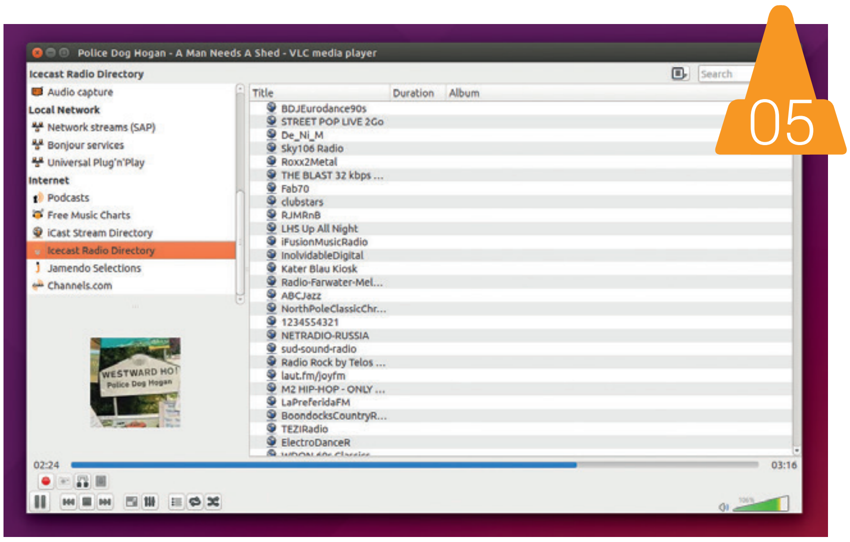
Device Selection

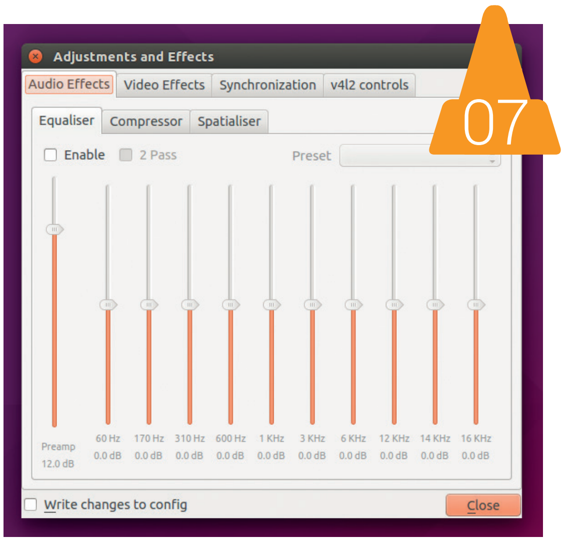
Your display will be opened and played in order to stream or save it.

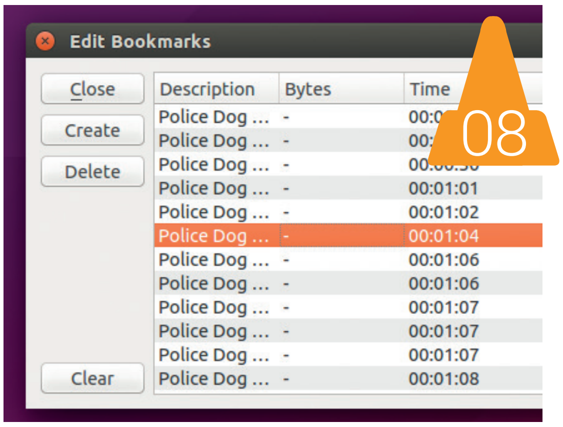
Options

Desired frame rate for the capture. 1.00 f/s









Description	Bytes	Time
Police Dog ...	-	00:0
Police Dog ...	-	00:
Police Dog ...	-	00:00:58
Police Dog ...	-	00:01:01
Police Dog ...	-	00:01:02
Police Dog ...	-	00:01:04
Police Dog ...	-	00:01:06
Police Dog ...	-	00:01:06
Police Dog ...	-	00:01:07
Police Dog ...	-	00:01:07
Police Dog ...	-	00:01:07
Police Dog ...	-	00:01:08

listen to podcasts (such as the fortnightly Linux Voice cast), go to View > Playlist, then in the internet section, click the plus icon next to Podcasts and enter the RSS URL for the cast. You can also use this section to listen to an internet radio station.

06 Record current video

You can record sections of the currently playing video by clicking on the record button (a red circle) in the control panel at the bottom of the *VLC* window. If you don't see the button, go to the View menu and click on Advanced Controls.

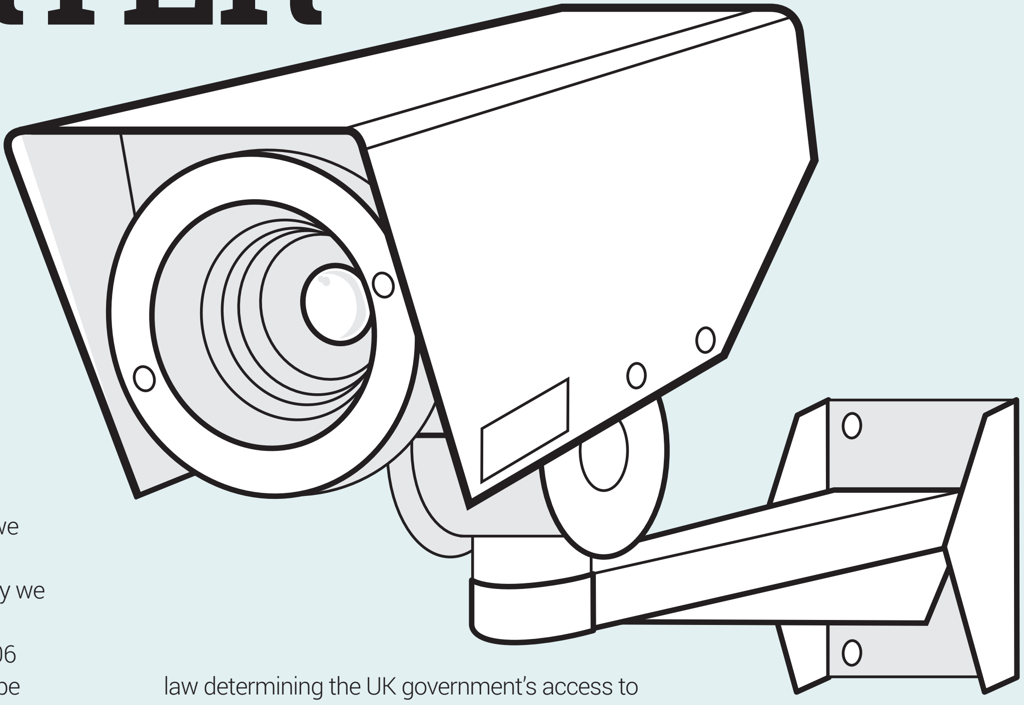
07 Audio and video effects

Sometimes, videos need a little treatment before they're good to watch. This could be some judicial cropping, noise removal from the audio or adjusting the sync between the video and audio. There's no need to reach from some complex video editor though, because *VLC* can do all this on the fly. Just go to Tools > Effects and Filters to set up whatever you need.

08 Bookmark a video

If you find yourself needing to go to specific scenes in a video frequently (perhaps you're watching *Gladiator* and need time to take in the magnificent 'husband to a murdered wife' speech), you can make your life a little easier by bookmarking particular points in the playback. Go to Playback > Custom Bookmarks > Manage to set up the bookmarks, and you'll be able to return to those points with a single click. There's no need to scroll back and forwards seeking out a particular scene again.

THE SNOOPER'S CHARTER



The UK government want to watch what its subjects do on the internet – wrongly, says **Ben Everard**.

Technology changes quickly. Law, on the other hand, changes slowly. The laws we abide by today are roughly the same as the ones we followed last year, and the year before; in contrast, if we look back 10 years, the way we use tech has changed hugely.

Take smartphones, for instance. In 2006 Google showed the world its first prototype phone, and the Android operating system hadn't even been released. 2006 was also the year rumours surfaced that Apple was working on a phone (which would be release the following year). Contrast that with 2016, when Google and Apple combined have over half the world market share for mobile phones.

> The UK Home Secretary says that we need a new law to govern surveillance on the internet, and she's right

Look at the web and a similar thing is true. In 2006, Facebook opened to the public after having previously been limited to academic institutions. YouTube, now the third most visited site on the web, was less than a year old and first caught the public imagination. 2006 was also the first year of Twitter.

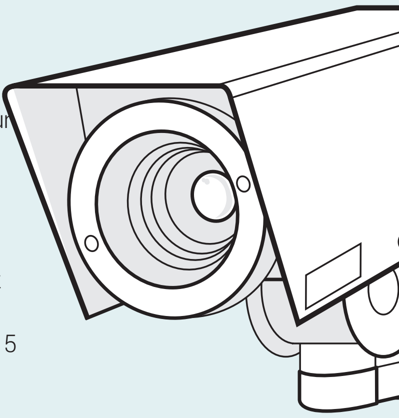
Go back a further six years and things seem even more distant. Only four of 2015's most popular websites even existed in 2000 (Google, Yahoo, Tencent QQ and Amazon). There wouldn't be a Wikipedia for another year. Windows 2000 and ME were the dominant new operating systems. Sun open sourced *OpenOffice.org*, though the 1.0 release wouldn't be out for another two years. Napster was still less than a year old.

This year, 2000, was also the year that the Regulation of Investigatory Powers act – the main law determining the UK government's access to our communications – was passed. It was written at a time when few could even conceive of a world where people routinely checked a collectively edited encyclopedia from their phones, and the idea of a social network hadn't yet germinated.

Theresa May, the UK Home Secretary, says that we need a new law to govern surveillance on the internet, and she's right. However, we need to be careful: any new powers granted to the state are likely to stay for a long time, and will apply even when our use of technology has changed even more than it has since 2000.

Who knows what tomorrow brings?

It's impossible to say what the future will hold, but looking back at the change from 2000 to 2016 and extrapolating forward, it seems almost certain that technology will become even more integrated into our daily lives in ways we can't yet comprehend. It's with this technology of the future that any new law will interact, so we need to ensure that it will enable us privacy and security in a future world in which the amount of data about us stored online will be almost inconceivably large. The draft Investigatory Powers (IP) bill released by the government in November 2015 gives the state sweeping new powers, but does it adequately protect our privacy?



THE BILL

What's in the new law that the powers that be want to pass?

If the draft IP bill came into effect, it would allow the state to compel "anyone providing communications services" to take "all reasonably practicable steps" to intercept the data of their users. Here there are two terms that have very loose definitions that will allow the bill to be applied in a huge range of cases. "Communications service provider" is a term that can be applied to just about anyone who has anything to do with any form of data exchange, from an internet service provider to an app developer, and "reasonably practicable" covers a lot of things that are easy to do but ethically wrong, such as deliberately switching to a weak encryption algorithm.

Dragnet!

The section of the bill covering bulk data collection is the most worrying aspect to us at Linux Voice. It allows the government to require anyone who handles communications to hand over vast swathes of data about people who aren't suspected of any wrongdoing. The explanatory notes accompanying the bill attempt to justify these with the following:

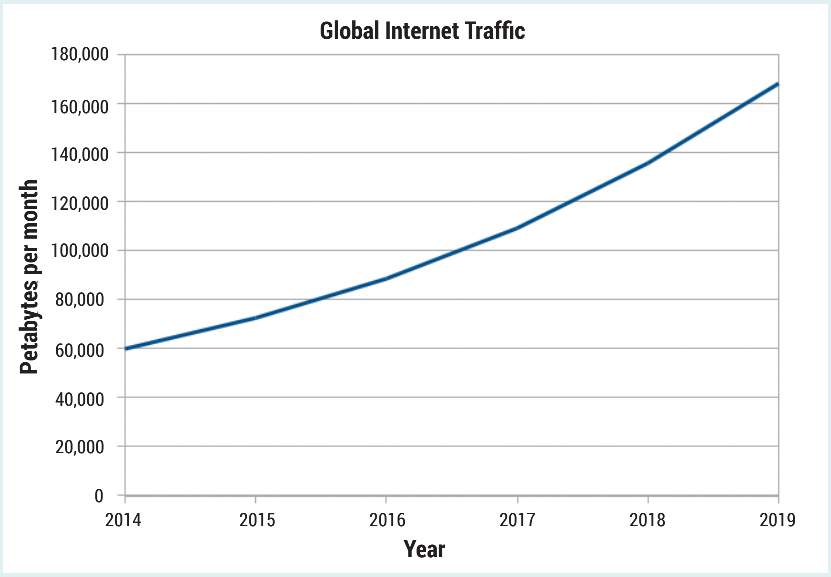
"Access to bulk data is crucial to monitor known and high priority threats but is also a vital tool in discovering new targets and identifying emerging threats. The law provides for the use of interception, communications data and equipment interference powers in bulk. These can be used to obtain large volumes of data that are likely to include communications or other data relating to terrorists and serious criminals. Robust safeguards govern access to this data to ensure it is only examined where it is necessary and proportionate to do so."

This is probably the most misleading passage in the notes. The first part of the first sentence is demonstrably false: access to bulk data is absolutely not crucial to monitoring known and high-priority threats, because if a threat is both know and high-priority (it can't be high-priority without being known), then there is no need for bulk collection – an individual warrant for the data on the known threat would equally suffice.

The above quote then points out that if they grab enough data, they'll probably get some relating to terrorists and serious criminals. While this is true, it is based on the logic that violating people's rights is acceptable if you violate enough people that you happen to also violate the rights of criminals and terrorists – also know as the 'if you arrest everyone, you're sure to arrest all the criminals' argument.

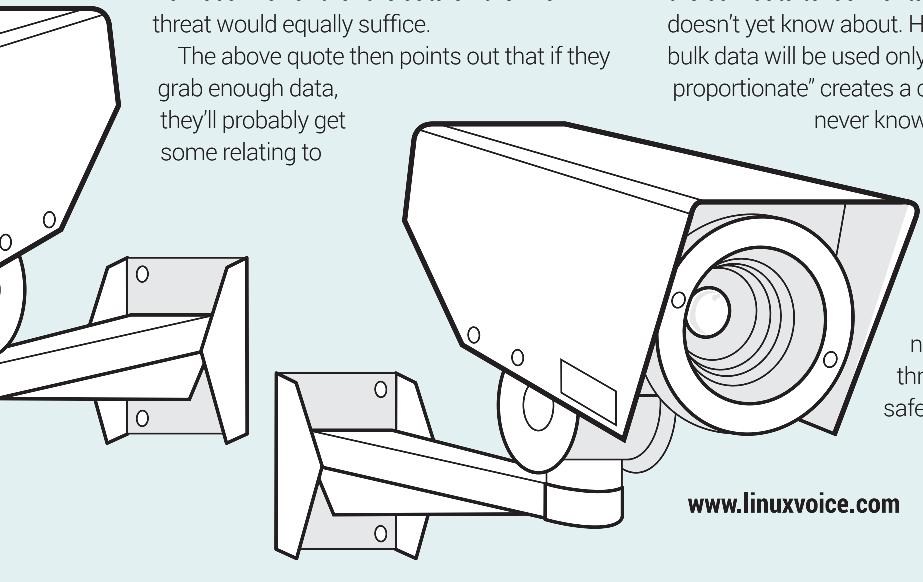
Perhaps the most revealing aspect of this passage, however, is revealed when you look at it as a whole. "Access to bulk data... is also a vital tool in discovering new targets and identifying emerging threats... Robust safeguards govern access to this data to ensure it is only examined where it is necessary and proportionate to do so."

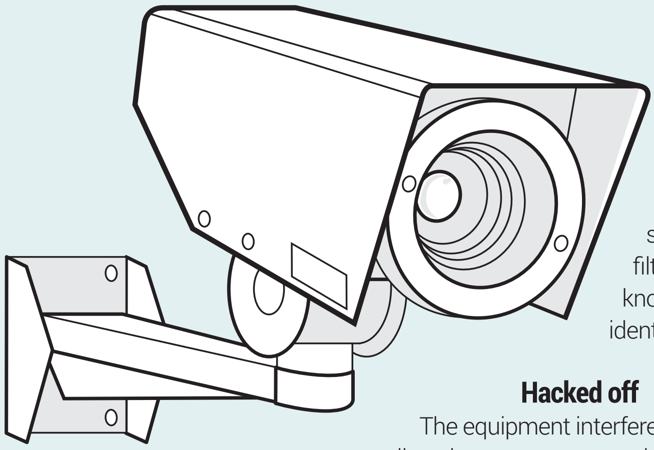
This implies that the government wants to use the bulk data to look for targets and threats that it doesn't yet know about. However, the assurance that bulk data will be used only where "necessary and proportionate" creates a contradiction, since you can never know when it's necessary and proportionate to examine data if you don't know the targets and threats that you're examining it for. The only way that bulk data can be used to identify new targets and emerging threats is if there aren't robust safeguards ensuring that it's



Global internet traffic is growing by 23% per year. If this continues, then any interception laws will become 23% more invasive every year.

Access to bulk data is absolutely not crucial to monitoring known and high-priority threats





only examined where it is necessary and proportionate. Any robust safeguard would ensure that security services only filtered the data for known targets and already identified threats.

Hacked off

The equipment interference aspects of the bill allow the government to alter either the software or hardware of your machines in order to extract information. In other words, it gives them the right to hack computers.

It may be useful to target individual equipment when done in accordance with proper safeguards; however, there is no situation where bulk equipment interference can ever be justified. It is never appropriate for security services to routinely hack the software and hardware of innocent people who aren't suspected of any wrongdoing. It was under a similar bulk warrant that GCHQ hacked into the computers of Privacy International, a charity dedicated to the right to privacy. At the time of writing, this hacking is under investigation by the Investigatory Powers Tribunal, because GCHQ didn't have the powers that it would have under the new bill. If GCHQ hacked into the computers of charities before they had the bulk equipment interference powers that this new bill confers, they're unlikely to back off when they have more powers to attack computers in vast numbers.

The government will be able to build up a far more detailed picture of your browsing habits than first appears

Another aspect of the bill requires ISPs to hold onto everyone's internet connection records (ICRs) for a year. These ICRs will detail every communication you sent, including web requests. This part of the bill makes a distinction between metadata (that is, data about the communication, including which machines took part and when it happened), and the contents of the communication. In the case of web browsing, this means that the ISP will record every site you requested data from, but not the actual page you requested. In other words, if you visit a BBC news page, the ISP will record a visit to **www.bbc.co.uk**, but not the story that you visited. The explanatory notes describe ICRs thus:

"An ICR is not a person's full internet browsing history. It is a record of the services that they have connected to, which can provide vital investigative leads. It would not reveal every web page that they visit or anything that they do on that web page."

This requirement for only the metadata means that sites can still be served over secure connections (HTTPS) without violating the law. However, it's actually a more nuanced situation than this, because a web page isn't a single entity that you download in one go: it's made up of images, scripts and styles that all come in different requests.

Each of these requests will go on your ICR, and by looking at the pattern of all these requests, the government can build up a far more detailed picture of your browsing habits than first appears. Depending on the website visited, our tests showed that this pattern of request was often sufficient to distinguish the category of page visited within the site, and sometimes the actual pages visited.

Left: Theresa May is the current Home Secretary, but the powers in the bill will allow any future Home Secretary to authorise spying of the entire populus. **Above:** GCHQ, based in Cheltenham, carries out much of the government's most invasive spying.

THE STATISTICS OF MASS SPYING

Bulk collection does not and cannot stop terrorism.

The current UK government is trying hard to convince the population that bulk collection is an essential tool in the fight against terrorism: that by collecting enough information, and by analysing it carefully enough, the security services will be able to work out who's a terrorist and stop them before they commit an act of terror. The problem with this is that it's mathematically wrong.

The first thing to take into account is that terrorists are rare. In the year ending in March 2015, 118 people were charged with terrorism offences in the UK. No attacks happened in the UK in the same period, so there can't be a significantly higher number of active terrorists than this. The UK population is about 64 million people, which means that, even if we play safe and scale up our estimate of terrorists by a factor of 10, terrorists only account for one in 64,000 people.

Now let's consider the algorithm that the state will use to process this bulk data and find terrorists. This isn't an especially easy problem, since terrorists will be attempting to blend in, and many ordinary people's curiosity takes them to websites about terrorism. The algorithm will never be 100% accurate, so what happens if it's not? What happens if the algorithm is, say, 99% effective at finding terrorists?

Enter Bayes' theorem

Bayes' theorem tells us that the probability of a person being both a terrorist and flagged by an algorithm that's 99% effective can be calculated by multiplying the probability of a person being a terrorist by the probability of a terrorist being flagged by the algorithm (0.99), then dividing the result by the probability of someone being flagged regardless of whether or not they're a terrorist (0.01). The result is 0.0015. In other words, the chances are that your theoretical 99% effective algorithm will actually be wrong 99.8% of the time. The reason for this apparent contradiction is because there are so few terrorists. The 1% of the time that the algorithm falsely flags a non terrorist as a terrorist (1% of 64 million) vastly outweighs the proportion of the time that it correctly flags a terrorist (99% of 1,000).

If this terrorist search filter were used as an input for a more involved investigation, the problems wouldn't get any better. With a success rate of 0.0015, the security services would have to investigate on average 667 people for every terrorist they caught. With the UK police officer numbers having declined by 17,000 (about 12%) between 2010 and 2015, the police don't have the manpower to chase up this number of inaccurate leads. GCHQ is expanding by 1,900 anti-terror spies, but even these numbers fall far short of the number required to make their way through the sheer number of leads that a 99% effective algorithm would generate. Even if all these leads were followed up completely correctly and every flagged terrorist were apprehended, the algorithm still missed 1% of terrorists, so this would leave 10 free to carry out an attack.

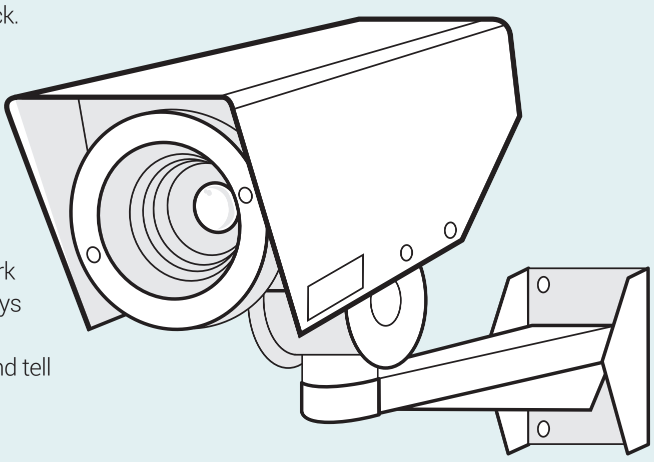
Not only does bulk collection not work, but it cannot work. Targeted, individual surveillance coupled with traditional methods of intelligence gathering and police work are the only effective ways of tackling the problem. Now write to your MP and tell them so! LV

Thomas Bayes developed his theorem in the 18th century, and it still has applications in bulk surveillance and spam filtering.

Safe spaces

Speaking in the wake of the Charlie Hebdo attacks in January 2015, David Cameron reiterated his desire to be able to intercept the entirety of online communications: "But the question is are we going to allow a means of communications which it simply isn't possible to read. My answer to that question is: no, we must not." The Draft Investigatory Powers Bill is the fruit of that desire. However, the Prime Minister's ambitions are not met by this new bill for two simple reasons:

- It places the requirement to intercept data on the telecommunications operator, but in the case of user-implemented end-to-end encryption (such as GPG), this operator has no capabilities to intercept the plain text.
- It only has the capability to force companies with operations in the UK to comply with warrants. Any non-UK citizen could set up an organisation outside the UK offer truly private communications without the need to comply with this law.



FAQ CHIP

The Raspberry Pi Zero isn't the only ultra-cheap single board computer in town. Say hello to the $9 CHIP.

MIKE SAUNDERS

Q Isn't a chip just a part of a computer? What's the deal with the name?

A We don't think it's a great name either – the CHIP has several chips on it, and searching for support on the internet is going to be tricky. Googling for "CHIP not working" is going to bring up lots of unrelated results, although it's not as bad as the situation in Gnome ("*Web* won't connect to the, er, web...")

But anyway. CHIP (or officially C.H.I.P., but we get a headache from seeing it written like that all the time) is an ultra-cheap, $9 single-board computer very much like the Raspberry Pi Zero. Indeed, it was announced and crowdfunded on Kickstarter back in May 2015, half a year before the Pi Zero was revealed to the world, and the team behind it managed to get over $2,000,000 from almost 40,000 backers.

Q Wow, that's not bad. Who's making it?

A Next Thing Co, a small startup founded in 2013 and based in Oakland, California. The company's first product was the Otto, a customisable digital camera that's powered by – of all things – a Raspberry Pi. By connecting the Otto to your smartphone, you can add filters, perform post-processing effects and do other tricks without having to be an expert in photography. You can even use the camera to take videos in animated GIF form, which is ideal if you're looking to build up sweet, sweet karma on Reddit's /r/gifs forum.

Q What is it that makes the CHIP so special?

A First and foremost, the price. We may all be cooing over the $5 Raspberry Pi Zero now, but back when it was announced, the $9 price point for the CHIP raised plenty of eyebrows in the wider computing world. Just the fact that a fully functional, usable computer (running Linux of course) would be available for under a tenner was brain-bending for many people. That's a real computer for the price of a couple of beers at the pub.

Part of the original Raspberry Pi's appeal was its low price. We don't want to sound wasteful, but there was something especially appealing about a computer that you can break, throw away and replace for a pretty small outlay. The CHIP takes this to the next level: want to use a tiny Linux-based computer for home brewing, monitoring wildlife or doing other tasks where it could end up broken? Buy a few CHIPs and if one buys the farm, just swap it out for another. (And go without the couple of beers at the pub that night to claw back the cash.)

> CHIP supports wireless networking and Bluetooth out of the box, whereas the Raspberry Pi Zero has neither of these

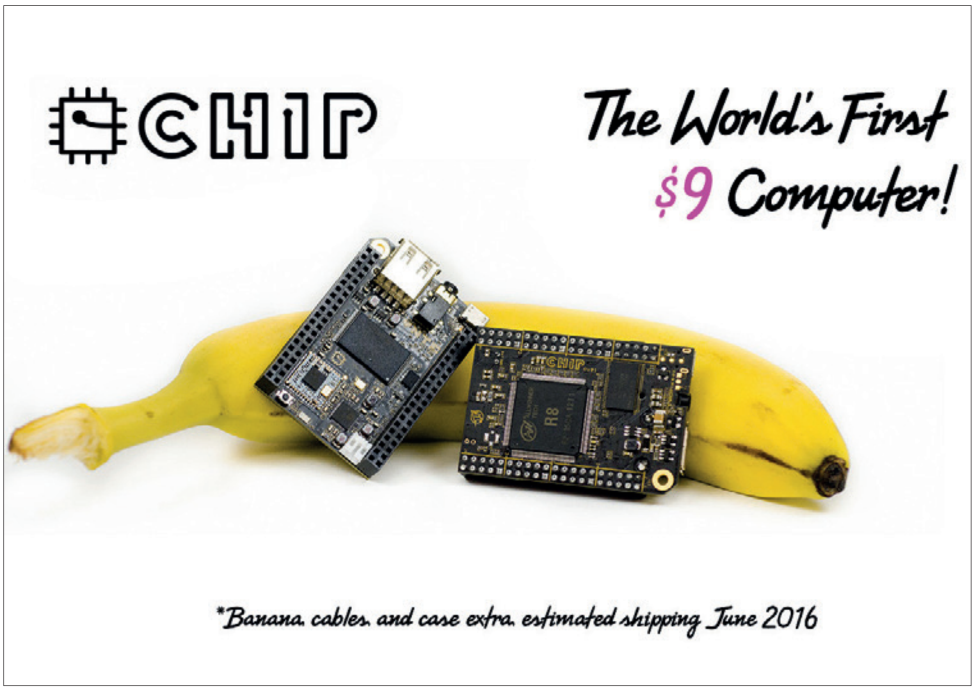
Q Fair enough, but the Pi Zero beats it in that respect now, doesn't it? Doesn't that mean the CHIP is obsolete?

A Well, yes and no. Both prices are astonishingly low, and it's true that the Pi Zero has the edge if you really want to save every penny. But there are also some substantial differences to the hardware. Both devices have single-core 1GHz ARM CPUs backed up with 512MB of RAM – but while the Pi Zero requires a Micro SD card for storage, the CHIP has 4GB of flash memory built in.

Similarly, CHIP has a single full-size USB port, in comparison to the Micro USB socket on the Pi Zero, so that's one fewer adaptor you may need to buy. And then the CHIP supports wireless networking and Bluetooth out of the box, whereas the Pi Zero has neither of these. So while the Pi Zero may look like the cheaper option at first glance, when you start to think of the adaptors you may need to kit it out with USB and Wi-Fi support, its price can match (or even exceed) that of the CHIP.

Video-wise, the Pi Zero has a Mini HDMI port (again, requiring an adaptor in most use cases) whereas the CHIP just has composite video with the option to convert to HDMI via another adaptor. And in terms of GPIO, the Pi Zero has the lead with 40 pins, in contrast to the CHIP's 8 (although this may change as the product reaches widespread production).

So both boards have their upsides and downsides. The CHIP is more capable out of the box, but the lack of a Micro SD card slot means you have to deal with the 4GB storage limit or start adding a USB flash drive or SD card reader – which again bumps up the total price.



You'll have to wait a few months to get your hands on a CHIP, but it will give the Pi Zero some friendly competition when it comes to embeddable computing power.

Q And what about those all-important shipping costs?

A At the time of writing, delivery of CHIP to addresses in the United Kingdom and United States costs $6.22 using the charmingly named Super Standard Shipping method. For the Pi Zero, our friends at Pimoroni charge £2.50 to post orders under £50 to the UK, or £4 to the US. So again, the prices aren't drastically far apart in this department, and you could save on the postage by ordering a bunch of CHIPs or Pi Zeros at the same time.

Q What versions of Linux run on the CHIP?

A Out of the box, the device will run Debian GNU/Linux. Next Thing Co is keen to position the CHIP as a general-purpose computer, capable of surfing the web, playing games and editing documents in *LibreOffice*. Given the computer's very modest specifications (and our experiences with the original Raspberry Pi), we think this might be overly ambitious – but we'll know for sure when we spend more time with it.

But outside of home desktop usage, the CHIP could be a great alternative for many of the jobs at which the Raspberry Pi excelled: robotics, home media servers, simple network attached storage devices, classic 8/16-bit console emulation, ARM assembly hacking and others. We look forward to seeing the CHIP hacked into old ZX Spectrum and Commodore 64 cases, running emulators and hooked up to televisions.

Q OK, I'm sold! How do I get hold of one?

A Well, you'll have to be patient. Next Thing Co is working on getting its first batch of shipments out to the Kickstarter backers – so those are taking priority. You can order a CHIP from the website at **www.getchip.com**, but you may have to wait until at least June before it will arrive in your letter box. In the meantime, Next Thing Co has promised to send a review unit to Linux Voice Towers, so we'll have a closer look at the machine next issue and you can start thinking about CHIP projects before yours arrives.

Q And what comes next? Is there going to be a CHIP 2? Or a CHIP Zero for 99 cents?

A Maybe one day! There are no public plans for a followup device right now, but the Next Thing Co team aren't sitting around twiddling their thumbs. The PocketCHIP (**www.getchip.com/pages/pocketchip**) converts the device into a fully fledged handheld computer, adding a resistive 4.3-inch touchscreen, miniature QWERTY keyboard, GPIO breakouts and 5-hour battery. You can order one now, but it costs more than five times as much as the CHIP itself at $49.

In addition, Next Thing Co sells various converters and add-ons such as a HDMI adaptor for $15, VGA adaptor for $10, composite video cable for $5 and a case for $2. As with the Raspberry Pi, we expect to see plenty of third-party add-ons and accessories once the supply chain for the device itself becomes stable and lots of people have the devices in their hands.

Q I don't know what to buy – tell me how to spend my money!

A Why not get both? Just like with KDE vs Gnome, *Emacs* vs *Vim* and Z80 vs 6502 (at least for us retro fans), there are zillions of debates concerning CHIP vs the Pi Zero raging on the internet. Teams have assembled on both sides trying to win over undecided buyers, but given the crazily low prices, we really don't see the need for flame wars.

Competition is good, so we hope this spurs on the development and release of even more ultra-cheap single board computers. Unlike with high-end smartphones or laptops, you don't just need to settle for one – build up a little arsenal of low-cost Linux-powered devices and share the love.

LESLIE HAWTHORN

COMMUNITY CURATOR

Graham Morrison meets an award-winning Free Software/open source advocate and community manager extraordinaire.

Leslie Hawthorn has been involved with open source for a long time. She spent five years as Google's program manager for its brilliant Summer of Code project. After stints at Red Hat, as its Engineering Team Manager, and the Oregon State University, as its Open Source Lab Community Manager, Leslie is now the Community Manager for Elasticsearch, the open source company responsible for the ultra-scalable search platform that powers search on sites like Wikimedia, StumbleUpon, Mozilla and Etsy. Her personal emphasis has always been on cultivating open source communities to create productive and inclusive environments for Free Software. And that's exactly where our conversation starts.

LV Do you think the open source community is becoming more inclusive?

Leslie Hawthorn: I think the community is slowly and surely becoming more inclusive, which I find to be quite good. There are certain debates that we were never having before that I think are important. The idea of, "Let's talk about a lack of women in tech," or, "Let's talk about a lack of people of colour in tech". This is not a new debate by any stretch of the imagination, or Anita Borg wouldn't have created the Systers email list, what, 30 or 25 years ago. It's certainly not new. One of the things that I find to be difficult is that because discussions are more prominent, they are much more politicised. If the dialog constantly is perceived by you as being 'you are a bad person because you fall into a privileged group', you're just not going to listen. You simply will not listen. Why should anyone feel bad for having had a good life? That's not a reasonable request to make.

I think it's more about changing the dialog from 'as a privileged person, you are an oppressor', to 'you are a privileged person, you are in a powerful position to be able to ensure that other people can enjoy the successes that you have'. One of the beauties of free and open source software is that you can give back, and this is another way in which you can give back.

LV It's becoming increasingly intimidating to get involved in the community – everything is out in the open now and your CV is online. Do you think this is a good place for us to have gone?

LH: I think that's tough. On the one hand, it's wonderful to be able to say your work is available publicly and you can stand on your own merits because everyone has access to see what you are capable of doing, and to some degree you can say that this removes

Leslie won O'Reilly's prestigious open source award way back 2010 for her work making Free Software communities better.

“People get over their intimidation factor a lot more quickly when they feel like they’re dealing with a human being who cares about their success.”

bias, because then people can objectively judge the quality of my work. And yet there is a whole lot of problematic stuff involved in the idea of saying 'well, I don't need you CV anymore, I just want to see what's available to you on GitHub'.

Not everyone has had the privileged position of working in a company in which they were encouraged to do open source software development. That's gaining traction and much more prominence now but that certainly has not always been the case. Depending on the organisation you work with, they may have a wonderful dedication to publishing open source software but that doesn't mean that all of their software is published as open source, so that should not diminish the quality of your work.

And then there's the question of how much time you have to devote to effectively writing software for fun or for a hobby or to scratch your own itch if that's not part of what you're paid to do during your day job. I know that I have commitments to taking care of family members, I have commitments to volunteer work that are not about coding projects but they are about helping more software developers. I don't necessarily have time to prioritise putting together cool sample applications on GitHub so everyone knows that I'm fantastic. I'm fortunate in that I don't have to worry about that, I have other ways in which I've built a reputation within the community that speak for themselves and not everyone is in that position. I think it's a double-edged sword.

LV Especially for people who want to get in who may not be versed in the way that it works.

LH: If you look at the rise in programs to help new people get into open source – so this is everything like the Google Summer of Code spin offs like Rails Girls Summer of Code, Django Girls etc – they are a phenomenal set of tools to help people begin the on-boarding process and get into open source software development.

On the other hand, if you also look at the (there's no nice way to say this) critiques of people who go through those programs... Python is considered to be an easy programming language, and I think it's considered an easy language because the Python community has very deliberately, from very early on, made a concerted effort to help everyone get into the tent and be successful. It's not because Python is easy: it's because we as geeks value things that are really difficult and really challenging and for which you have to be a super genius, because that's part of how we define ourselves, and there's a kind of mismatch there.

> Fundamentally, geeks are profoundly insecure; we just don't want to admit it

Fundamentally, geeks are profoundly insecure; we just don't want to admit it. And part of what drives us and our success is that insecurity. I think it's OK to be insecure; just own that insecurity, know what it is, know how it acts and operates in your life, and don't let your tools use you. But if it's a tool to become more ambitious, more successful, to learn new things, then great – that's wonderful. If it puts you in a position where you need people to be less successful than you in order for you to feel good about yourself, that's when it becomes problematic. If you're competing with yourself, that's one thing. If you're competing with everyone

A San Fransisco Bay area native, Leslie has upped sticks to beautiful Amsterdam, the home of ElasticSearch BV.

Outreach programmes such as Google's aren't enough to foster a more inclusive environment in Free Software – a deeper cultural shift is required.

else in situations where it's not really a competition, it's a situation where a rising tide lifts all boats, that's when it becomes problematic.

LV So to give people the toolkits to improve their communities and make them more approachable and less intimidating, what should those communities do? More outreach programs like Google's Summer of Code?

LH: I think Google's pretty much alone doing projects like that for a couple of reasons. One, Google being Google, people figure 'OK, Google's got it': Google's got my email, Google's got my documents.

And I don't think that outreach programs like that are the sole means to make communities more welcoming. I also think it's as simple as making your documentation approachable and also inviting people who don't have expertise in your project whatsoever to go through your documentation, because if somebody is not able to effectively do the task that is documented, it's not because this person is foolish, it's because the documentation is broken or, potentially, the software is broken.

LV But the documentation is nearly always broken.

LH: Exactly. There's a gentleman by the name of Rich Bowen who is a docs guy for the Apache Software Foundation. I just ran across a presentation he did and it was so incredibly good about the importance of documentation to your free and open source software project.

About two slides in, when he's talking about DocBook and what not, he talks about the importance of empathy and creating good documentation, being able to see things through the eyes of your user. He also says that the response 'RTFM' is the height of arrogance because, one, you're assuming that someone hasn't read the manual, which isn't fair because often documentation is broken, and you're also making a whole huge load of assumptions about what someone has as knowledge in their heads.

One thing that that I think that we forget constantly, and this was brought home to me very sharply during my time running the Google Summer of Code program, we tell people just Google it and you can find the answer. If it's a specific error message then, fine, sometimes that will work for you, but how well are error messages written? [much laughter]. Someone who is not intimately familiar with your project – to the point where they know all of the gaps that you've already forgotten because you've created workarounds in your head already – they don't know what to search for.

LV So a new user or contributor might have no knowledge or context of anything. Being in that situation is awful because you don't want to look ignorant, like you've missed something on the FAQ, or whatever people will say.

LH: Absolutely. There are a couple of different things that I've seen be very successful and there are arguments for and against each of these tasks, but there are some projects, like the Drupal world where they have a channel that's for Drupal newbies if hanging out in the regular Drupal channel is difficult for you either because it's noisy, because there's several people on there on any given day, or it's difficult because you're intimidated because the creator of the Drupal project is in there and maybe you don't want to ask the silly question in front of [him]. Fair enough, then maybe you can ask in the Drupal Newbies channel. And there's also always the encouragement that it's great that you're in here, you can also go over there too, you can play on *Drupal*, it's cool.

Other projects have a human who is dedicated to be the welcome wagon, so if someone new turns up on a mailing list or IRC or the issue tracker, and they're asking questions that are clearly well thought out but under informed then it's their responsibility to welcome this person [and let them know that] 1) their ideas are valued, and 2) they're not dealing with a bunch of people who were sprung full born from the head of Zeus knowing absolutely everything (because that's not actually a thing).

Amazingly enough, people get over their intimidation factor a lot more quickly when they feel like they're dealing with a human being who cares about their success instead of dealing with a wall of super geniuses sort of

"That Thomas Edison quote: 'I didn't fail 10,000 times, I found 10,000 ways that didn't work on the way to making it work' – that's on posters for a reason."

staring down at them saying 'why are you wasting my time'.

And the same thing could be said of failure in all of those projects and workplaces.
LH: Absolutely. And the thing that I would like to tease out of these examinations of failure is that a good leader's job is to help the people around them get through their failures. If you are a capable competent human – and really all colleagues are capable competent humans – you don't need your manager to help you be successful. You don't, you can go succeed on your own.

You all need to get together and form a plan about what needs to happen, but it is not your manager's role to help you succeed. It's your manager's role to help you when you have failed to figure out what that graceful recovery looks like, and also to help you realise that the fact that this issue has come to pass is not the end of the world, it is not a reflection on you, it is a reflection on the outcome of that situation. OK, so you made a mistake, OK so you failed; well that's fine, what did you learn from it? If you didn't learn from it, here are some of the things I learned from it, perhaps I should not assign you a deadline two hours in the past, for example. Again, people don't need a lot of wisdom, hand holding and hugs when they're doing well. They need it when they're doing poorly. I think as human beings we're neurally geared as well as financially incentivised to focus on all of the bad parts about failure, as opposed to what we get out of it, which is new knowledge.

People have been thinking about this for a very long time.
LH: Yes, and clearly it's easy to say and hard to do, as with many good things. If it were easy to do, everyone would already be doing it.

We're moving into an era where we have a wonderful set of guidelines, such as the Agile manifesto or DevOps methodologies, you know: fail fast/recover quickly; make it graceful; it's not when your systems will fail or when you as a human being will fail. And that's all lovely and we pay lip service to all these ideas, but then we still operate in these ways that clearly subvert that as an idea, such as telling all of your team that it's totally OK to fail – that's not a problem, we just fix the problem, and then, instead of that actually being the way things are executed, instead we micro-manage every bit of the process and so people get the same message, which is that's it's not OK to fail.

And I also have a great case study in booking.com. So, my partner was a former site reliability engineer there and still has great ties there, and I have friends there because, you know, the ex pat life. They actually find that 19 out of 20 of their deployments that are customer facing have a net negative impact on revenue, sometimes into the millions of dollars over the space of 5–10 minutes before they're rolled back, and that they have a great deal of tolerance for that risk because the 1 out of 20 deploys that actually have a positive impact on revenue can move the needle in terms of millions or billions because they have found that one thing that actually works.

That's a cultural thing. Your theme mostly seems to be about empathy, because empathy is something you can't put in a spreadsheet.
LH: Absolutely. Interestingly enough, I will be giving a talk at Eurocamp with a wonderful gal named Dajana Günther who is on the board of Ruby Berlin, which is the non-profit organisation that looks after all of the Ruby activities in, obviously, the city of Berlin, but they're also highly involved in Rails Girls Summer of Code and it's all about empathy and why this is a valuable skill, but also how to cultivate empathy if you don't necessary think of yourself as terribly empathetic.

How can you cultivate empathy?!?

LH: This is terrific! This is actually something you can learn, so this is great because just telling people to be empathetic is not terribly helpful. I have a personal pet peeve against going to talks and people are like, do this, and give no information on how. So there are actually a couple of great ways to cultivate empathy, one of which is fairly easy for us, which is read fiction.

A leader's job is to help the people around them get through their failures

Really?

LH: Researchers have demonstrated that reading fiction helps you to be more empathetic because the process of fiction and actually going through the cycles of imagining the scenario in your head, while you're consuming the text, causes you to go through the thought exercise of, "What is this character experiencing?". I can empathise with that because I'm internalising it – use those same lessons you learn while reading fiction and apply them to the real world [*Reading Literary Fiction Improves Theory of Mind*, David Comer Kidd and Emanuele Castano, **www.sciencemag.org/content/342/6156/377.abstract**).

There's also some other cool scientific studies where people have noted that empathy is a choice and that people often chose not to be empathetic because they're not financially incentivised to do so. The interesting thing that rolls out of that is the need to create team structures, organisations or structures of any kind that specifically incentivise empathetic behaviours, because clearly as humans we are hard-wired to do this, but again, it needs to be learned and we need to have it proven that it will benefit us.

Do you think recent inclusivity issues within the open source community are unique to the personality types it attracts?

LH: Yes and no. The challenges to empathy are not unique to the Free Software/open source world at all. I think they are the result of increasing pressures on all social systems period.

I think we can just blow this out to the whole world and go, wow, there's now seven billion+ people, if you have concern over your livelihood you will tend to be less empathetic. One of the things that I think is a unique challenge for our domain is that we have a rhetoric of openness, inclusivity, meritocracy. Anyone can come and participate in this process, and that is true up to a point. There are certainly no rules saying that you may not participate. But, because we have that as our rubric for understanding participation in FOSS, we don't realise that that in itself is a problematic statement to make. It's not as simple as [assuming that] everyone is able to do this. There are various types of privilege that mean people are not able to engage, so it's easy for me to say that it's easy to succeed in the free and open source software world. I was fortunate to have a mother who worked for one of the large telecommunications firms. I had access to the command line when I was three, and I'm (now) almost 40.

And you used the command line at the age of three?

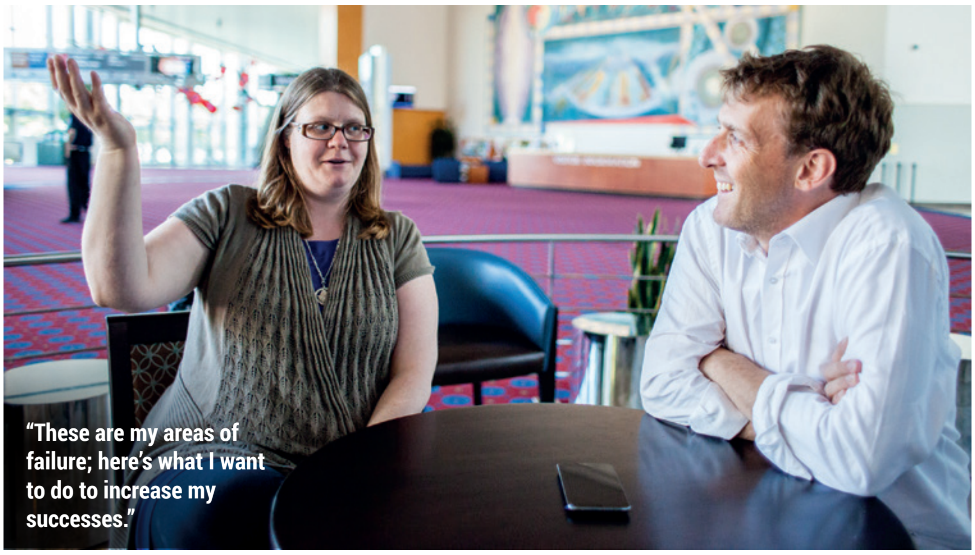
LH: Oh yeah! I think I was a strange three-year-old, and I stayed strange! It's very easy to think, "Well, I succeeded so anyone can succeed." And we like to think of ourselves as being special but not the special snowflakes, right?

We're special human beings because we have this great gift, which is the FOSS ecosystem, both in terms of the great tools we can use and the great community we have. And yet none of us wants to think that maybe we were deeply successful, not because we were super awesome but because of the tools we had to be successful with that no-one else had. That's not a comfortable position to be in.

Frequently people look at meritocracy, and meritocracy is a very comforting concept, because it means that I have succeeded because I am terrific and it's very hard to disassociate the notion of that you are terrific and you also have more advantages that allowed you to succeed. That's not a set of ideas that people are very comfortable holding their minds on.

It requires a personal level of abstraction that a lot of people aren't equipped for.

LH: Yes, and self awareness is hard, just in general. And I think we're not taught exercises in self awareness frequently [enough]. It's empathy for oneself and one's willingness to be humane towards oneself rather than simply brush all of our feelings under the carpet and pretend they don't exist. And honestly, empathy starts with the ability to show care for the self and really understand: these are my areas of success; these are my areas of failure; here's what I want to do to increase my successes; mitigate the places where I'm not doing as well as I want to do. Until you're able to do that with yourself, being able to provide that as a baseline of compassion for other people is, well, I won't say impossible, but it's certainly much harder.


"These are my areas of failure; here's what I want to do to increase my successes."

LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM

BUY LINUXVOICE MUGS AND T-SHIRTS!

GNU/
LINUXVOICE
FREE SOFTWARE FREE SPEECH
'PLAYFULLY DOING SOMETHING DIFFICULT, WHETHER USEFUL OR NOT, THAT IS HACKING.'
Richard Stallman

shop.linuxvoice.com

REVIEWS

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

Andrew Gregory **is 82.3972223% of the way through Dry January. But who's counting?**

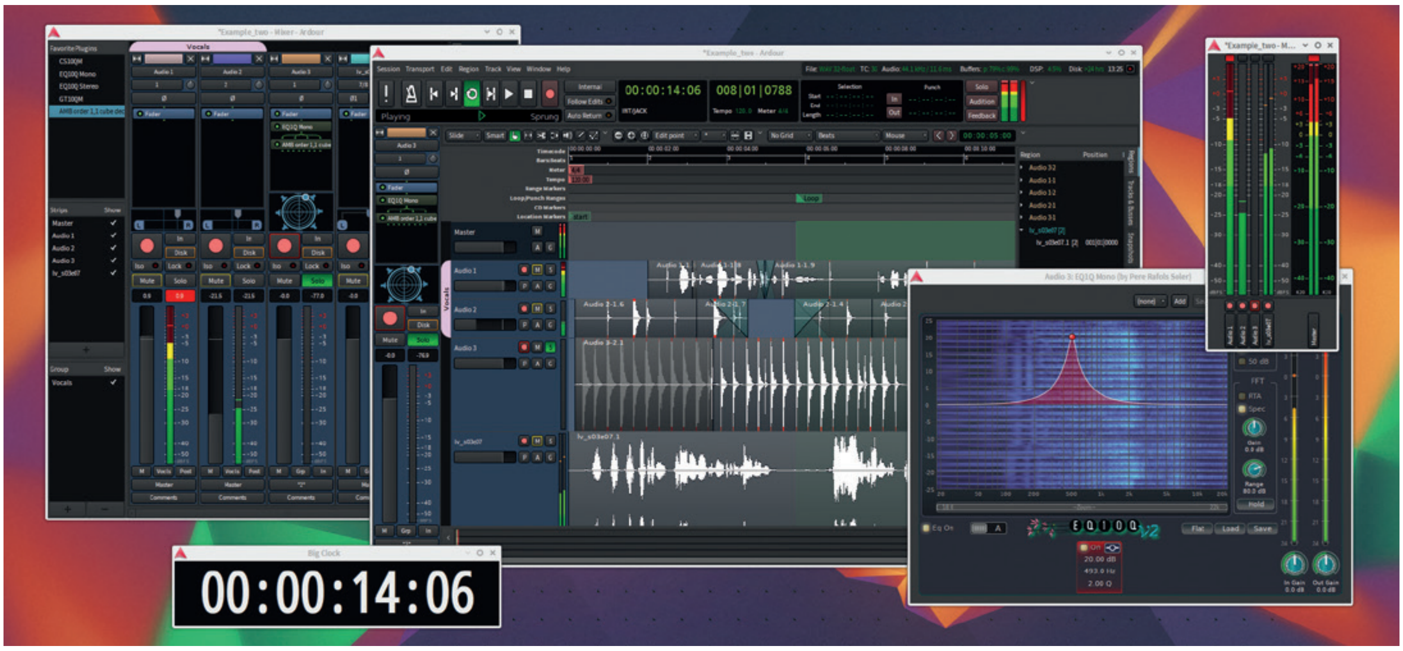
It's a new year, so let's make some predictions! I predict that Apple will release a cable for its devices that won't fit any of its old devices, so 'forcing' its customers to buy a load of new stuff to replace the perfectly good old stuff. This will take place, ooh, in spring. March maybe.

In August Microsoft will launch a partnership/foundation/call it what you will, with an organisation nobody's heard of, that purports to represent Free Software standards. This organisation will conclusively prove that MS loves Linux. Again.

A company with the info security of a wet paper bag will leak a load of customer data some time in October. This will be blamed on 'hackers', and not the use of passw0rd123 as their admin password. And more car makers will be sucked into the emissions scandal, as we pretend that the combination of the human nature and closed source software makes this anything other than inevitable. Meanwhile, unnoticed, Linux will continue to power everything from fridges to the internet. Carry on!

andrew@linuxvoice.com

On test this issue . . .



Ardour 4.6

Graham Morrison finds another excuse to turn on his immense synthesizer collection.

Web http://ardour.org
Developer Paul Davis
Licence GPLv2

We don't always agree with the monetising methods of *Ardour*'s author, Paul Davis. At the moment, for example, it's impossible to download a binary of *Ardour* from its main website without going through several nag screens, paying a one-off (low) fee or buying a subscription.

It's completely within Paul's rights to do this, and there's nothing stopping anyone downloading the code from GitHub and compiling the packages themselves (as we did). But binary obfuscation isn't going to help *Ardour*'s cause. We don't think making something harder to access is going to improve Paul's profits, especially when Linux distributions are going to bundle their own versions anyway and OS X or Windows users will just pirate something else. We humbly suggest that *Ardour* needs to get more, rather than less, exposure, and another solid update like this is a great advertisement.

Ardour is a multi-track audio editor. Unlike *Audacity*, it's designed to record and mix multiple recordings together, much as an engineer would in a recording studio. We use it to record our podcast. It's brilliant, and recent releases have taken the application from a difficult niche to what we feel is imminent mainstream recognition.

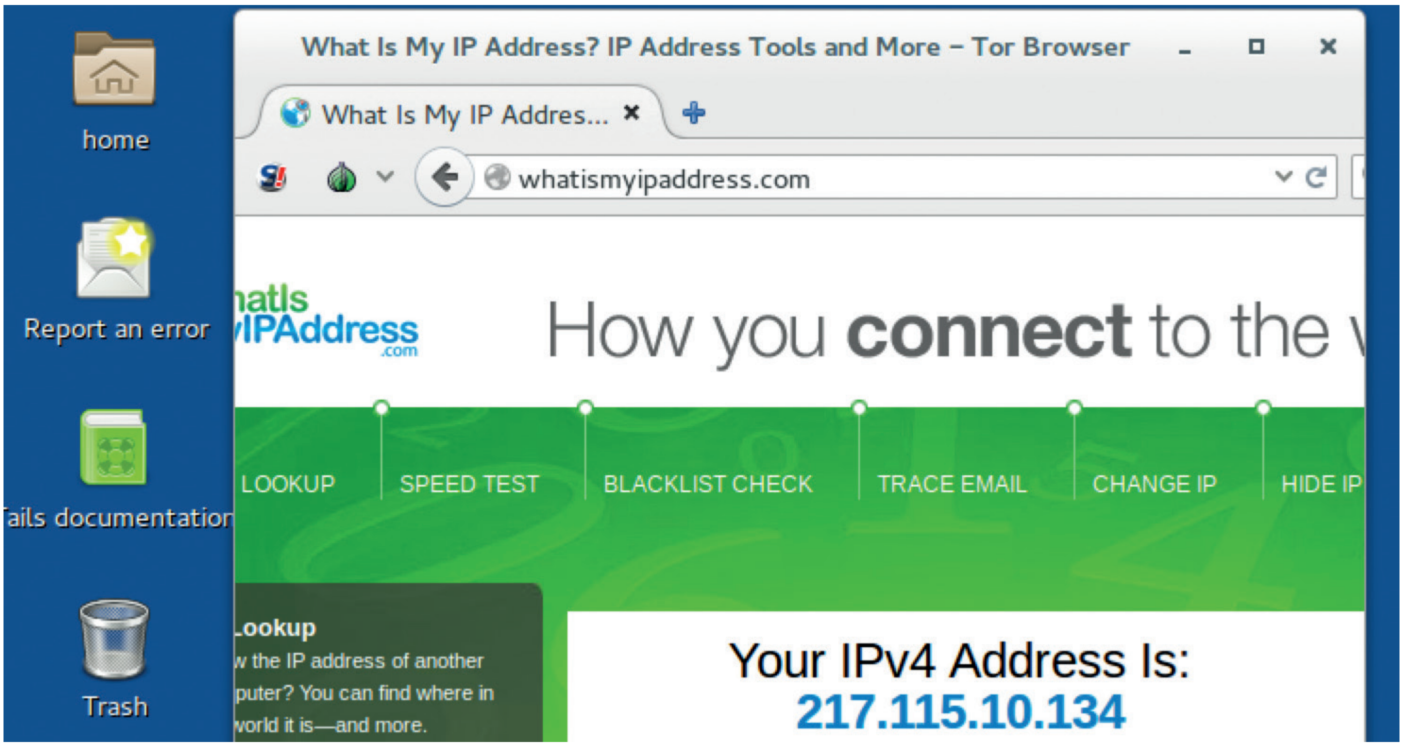
4.6 is a huge step forward too, mostly in the way *Ardour* handles plugins. These essential components, like filters in *Gimp*, enable you to process audio, changing the perceived volume of a track, add echo, or distortion, or whatever else the plugin developer has imagined. There are hundreds available for Linux and they're fundamental to *Ardour*'s functionality – we can't believe *Ardour* doesn't include its own compressor, gate and equaliser, for instance. But these external plugins are now a lot easier to use. The main mixer view even lists your favourites, right where you can drag them into your audio tracks. You can even drag and drop presets too, enabling you to quickly save and retrieve your hard-tweaked parameters. The mixer view in general is now 25% smaller, and the GUI has had lots of minor tweaks to make things look better. We also appreciated the ability to turn off all plugins when loading a project, as some plugins can cause a project to crash. Most similar applications have had this feature for years.

Ardour is developing at an incredible rate, and the quality of each release is staggering. All we can add is that if you use *Ardour* professionally, it's entirely worth spending your money on.

This release also has support for a major external controller, the Presonus Faderport.

Paul Davis and the *Ardour* team are doing an amazing job, creating a powerful open source audio editor to rival *Cubase*, *Logic* or *Pro Tools*.

★★★★★



Tails 2.0

A new release of this Tor-based distro keeps **Mike Saunders** safe online.

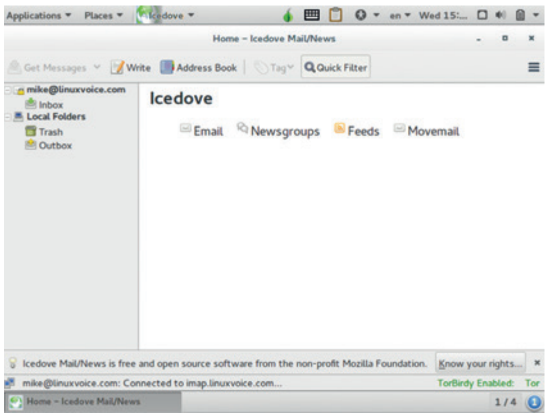
Tor is a fascinating project. It aims to provide a certain level of privacy and anonymity online by routing traffic through a network of thousands of relays scattered around the globe, but it's being attacked from all sides. Sure, there may be a few bad eggs using *Tor* for nefarious purposes, but it helps countless people in not-so-democratic countries communicate and get access to information that would normally be blocked.

Installing *Tor* manually can be rather fiddly, so one solution is the *Tor Browser*, a pre-packaged bundle of *Firefox* and *Tor*. Just launch it and start browsing – it's then very difficult for websites you visit to determine who you are. But still, if there's other unsecure software on your system (especially if you're running Windows), using *Tor Browser* alone isn't enough.

Here's where Tails comes into play. "The Amnesic Incognito Live System" is, as its name suggests, a live Linux distro that routes all its network traffic through *Tor*. Because it runs in live mode from a DVD or USB key, it leaves no trace on your hard drive (unless you explicitly choose to save files manually), so you can boot up a PC with it, do your work, and then power down the machine as if it hadn't been used.

So, what's new in Tails 2.0? The biggest change is the switch to Gnome 3 as the desktop. Sensibly, the Tails team is using the "Classic" mode of Gnome, so that it looks and behaves more like a traditional desktop. *Claws Mail* has been junked in favour of *Icedove* (an unbranded version of *Mozilla Thunderbird*), while the *Tor Browser* itself has been updated to version 5.5a6.

We've always been fans of Tails, as it really focuses on making privacy and security accessible to the masses. This new release upgrades the distro with a better desktop environment and email client (nothing against *Claws*, but *Thunderbird* is a better choice for less technical users). With certain governments desperate to monitor everything we do and put back doors into encryption, we need distros like this.

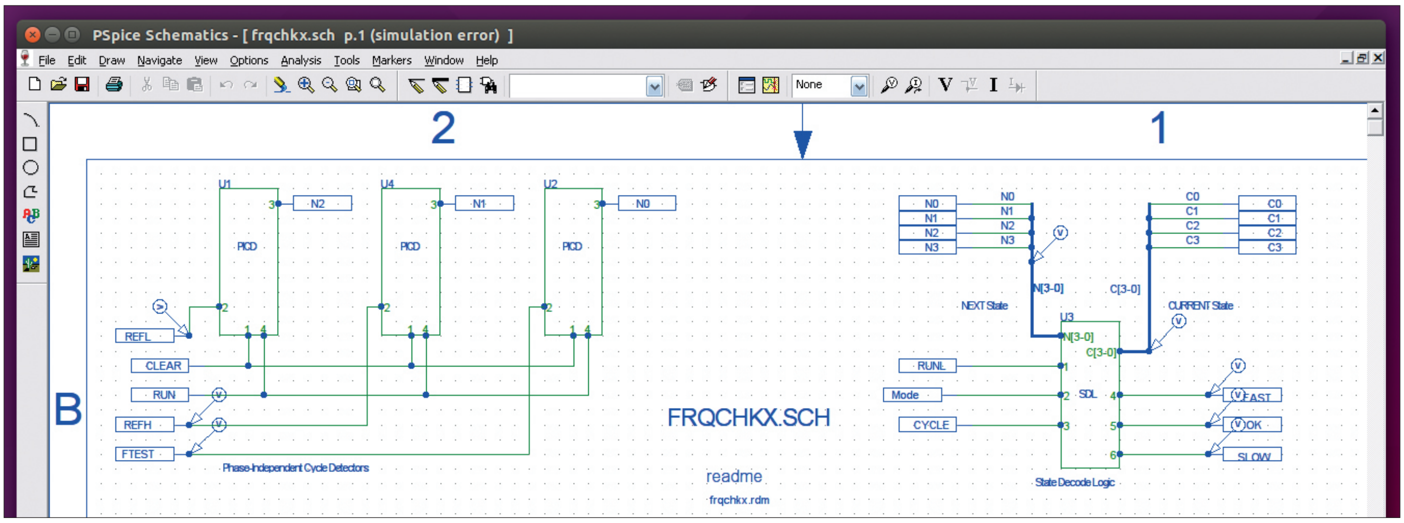


While *Tor Browser* is the star of the show, there are other programs included for communicating securely online, such as *Icedove* (a rebranded *Thunderbird*).

Web https://tails.boum.org
Developer Tor and Tails teams
Platforms IA32

The switch to Gnome 3 and *Thunderbird* keeps Tails fresh, modern and ideal for private browsing.

★★★★½


PSpice Schematics - [frqchkx.sch p.1 (simulation error)]
File Edit Draw Navigate View Options Analysis Tools Markers Window Help
FRQCHKX.SCH
readme
frqchkx.rdm
Phase-Independent Cycle Detectors
State Decode Logic
REFL
CLEAR
RUN
REFH
FTEST
RUNL
Mode
CYCLE
FAST
OK
SLOW

Wine 1.8

Ben Everard messes about with some Windows software – all in a good cause!

Web https://www.winehq.org
Developer Wine Authors
Licence LGPL

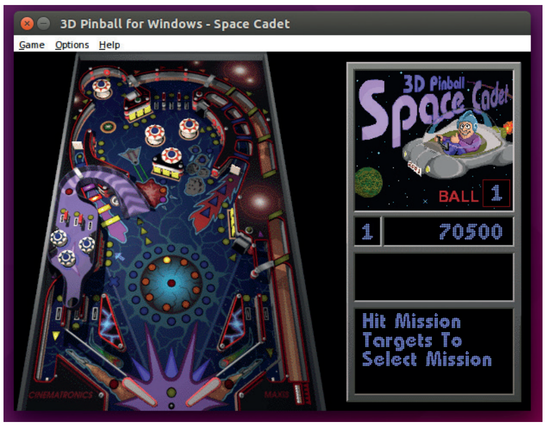
The name *Wine* originally stood for Windows Emulator – however, this changed when it became clear that the project wasn't, and wouldn't become, an emulator. The project kept the same name, but switched to the recursive acronym Wine Is Not an Emulator. Instead, it's a compatibility layer that enables the user to run Windows software on Linux.

At its heart, any piece of software (whether it's compiled to run on Linux or Windows), is just made up of machine code instructions. These are for the CPU and don't care about the operating system running. The only compatibility problems arise when this machine code tries to access the library of code that the operating system supplies, as these libraries differ between Linux and Windows. *Wine* is an attempt to recreate the Windows libraries (and other APIs) in Linux so that executable files for Windows will run in Linux. The problem for the *Wine* developers is that this set of libraries is huge, complex, poorly documented and changing. *Wine* is incomplete and probably always will be, but it doesn't have to be perfect: it just has to be good enough to run the software you want to run, and each release, *Wine* gets a little closer to matching Windows, and so more and more software runs. You can find out whether a particular piece of software runs or not by looking at the application database (**https://appdb.winehq.org**), but bear in mind that this is based on people's experiences and may be wrong or out of date.

Perhaps the biggest challenge for anyone using *Wine* is working out how to configure it for each application. This can be a significant challenge, and the simplest way around it is to use *Play On Linux* (*POL*). This contains configurations for many popular applications (as you may guess from the name, many of them are games). *POL* uses known-good versions of *Wine* for each application, so you won't immediately benefit from *Wine 1.8* using this until the configurations are updated.

The development of *Wine* never stops, and version 1.9 is already out, however, this is a development snapshot rather than a stable release. The next main version will be 1.10 which is scheduled for release in December 2016.

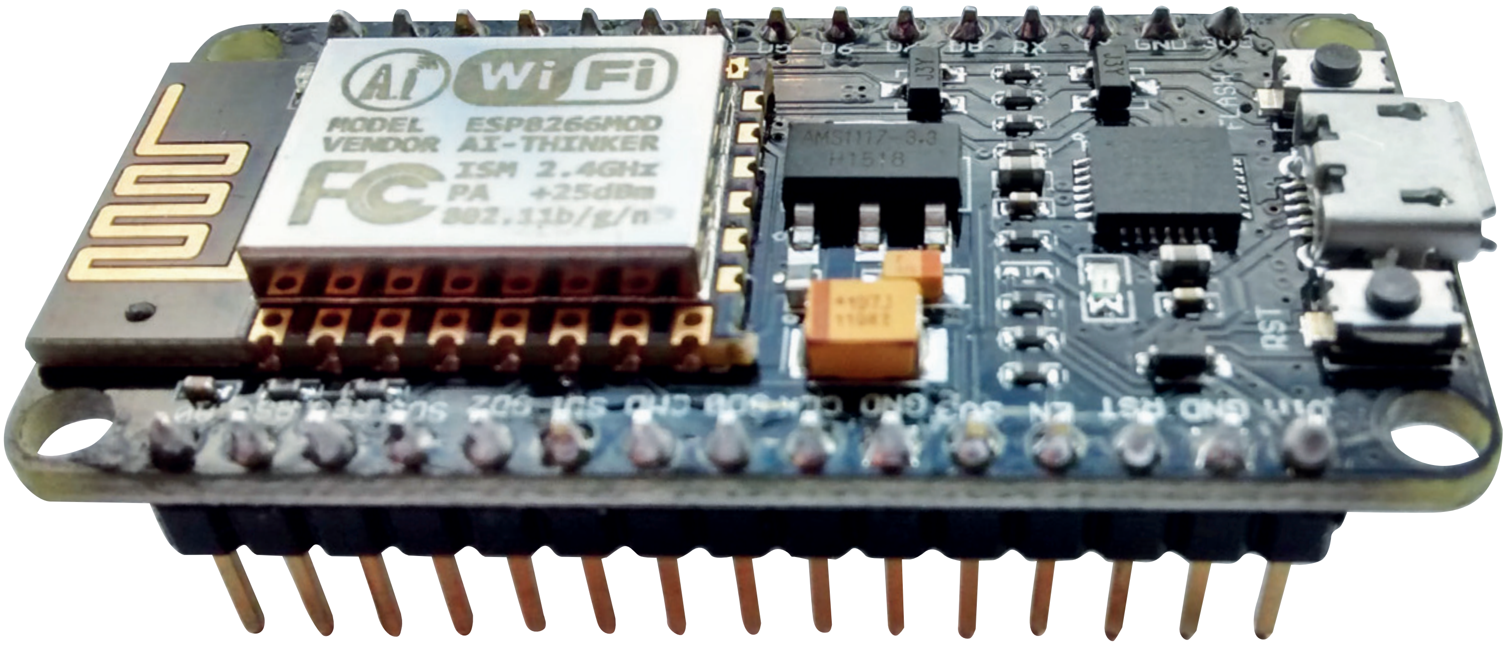
Wine 1.8 is the most complete release to date, and well worth an upgrade if you need to run Windows software. However, until the configuration is easier, we can't recommend vanilla *Wine* for regular users.


3D Pinball for Windows - Space Cadet
Game Options Help
3D Pinball Space Cadet
BALL 1
1 70500
Hit Mission Targets To Select Mission

With *Wine* you can enjoy classic Windows games such as Microsoft's *Pinball* on Linux.

An excellent way to run Windows software without having to dual boot, but let down by complex configuration.

★★★½☆



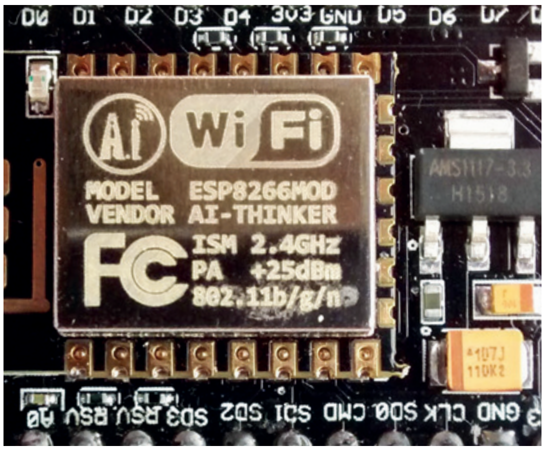
NodeMCU

Ben Everard tests out a small board with big ambitions.

A computer with Wi-Fi for £3 including tax and delivery might seem like an impossible dream – after all, most USB Wi-Fi dongles cost more than this. However, the NodeMCU really does deliver this as promised with one caveat: it's not a fully featured desktop computer, but a microcontroller. Don't expect to be able to run your favourite Linux distro on this, or any other Linux distro for that matter, but you can program it to run a single application.

The primary way of interacting with the NodeMCU is via a Lua command line interface that you can access over the USB port using either standard serial port software (such as *screen*), or specialist software designed for this module such as *ESPlora*. Using either of these methods, it's possible to program your NodeMCU to send and request data over the network. Coupled with the networking functions are eight digital input/output pins, one analogue output, an SPI bus and an I2C bus. To make using the module easy, version 2 of the board comes with a USB port for power and communication, so no other hardware is needed to program the device.

The connectivity, small size and low cost of the NodeMCU make it ideal for building internet-of-things devices. For just a few pounds more, you could buy a whole host of sensors to keep track of almost any aspect of your environment. The microcontroller is perfectly capable of running a web server to provide an interface to any device you can connect it up to.



The ESP8266 Wi-Fi module provides all features of the NodeMCU. Both the processor and the Wi-Fi connection come from here.

The downside to the NodeMCU is that it's still quite new and not as popular as some other microcontrollers (such as the Arduino), so there's not much other hardware designed to work with it at the moment. There's also not much useful documentation for people new to microcontrollers. However, if you have some experience with similar devices and are looking for a Wi-Fi platform, the NodeMCU may be the best option available.

Web www.NodeMCU.com/index_en.html
Developer NodeMCU team
Price From £3

A price and performance that is unequalled in wireless microcontrollers, but better documentation for beginners is needed.

★★★★☆

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

PASS ME THE WINE

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

Once a must-have for Linux gaming, *Wine* has become increasingly redundant of late. This is not just due to the increasing number of native titles available, but also to the lack of DirectX 11 support, which is now essential for most AAA titles.

While the compatibility layer is still excellent for older titles that are unlikely to ever get Linux support, such as *Skyrim*, gamers are going to be bitterly disappointed if trying out something like *Fallout 4*, which doesn't support DirectX 9. There is simply no way to run such games, let alone well, and with the release of DirectX 12 right around the corner, Wine is lagging behind.

Support for the API has been in "experimental" stages for a while now, and developers have been working hard to address this elephant in the room. That said, once support is available, we're unlikely to see a bunch of DX11 games working overnight. With this in mind, the software has been taking more of a "legacy" role for older titles.

While a cross-platform API such as Vulkan (successor to OpenGL) would be the ideal long-term solution, and is likely to gain traction over DX12 (if only for its cross platform support) this will also take a while to become an industry standard.

With this in mind, *Wine* is unlikely to disappear any time soon, since easily portable AAA games are still a while off, and there will always be a demand to play classic titles of yesteryear such as *The Sims* without the need to install Windows.

Divinity: Original Sin – Enhanced Edition

A true overhaul of a classic formula.

Many games have claimed to bring the dated CRPG genre into the 21st century, but few have really followed up on those claims as well as *Divinity: Original Sin*. From its fantastic fully-3D graphics to its usage of full voice acting, it is clear that the game doesn't simply seek to cash in on nostalgia.

This is most evident from the incorporation of controller support, something that previously seemed impossible to do in such a game without dumbing it down significantly. Though there are still needlessly clunky inventory menus found in CRPGs, navigation is streamlined and easy to use. The drawback of using a controller though is that text boxes get enlarged during dialogue and cover the whole screen, killing much of the immersion – something that seems completely redundant given the full voice acting.

Though the story follows many of the usual fantasy tropes involving dark guilds and plots to doom the world, it doesn't come off as overly dense given the presence of lighthearted gags. Similarly, the game at no point bombards the player with countless text boxes describing banal details, and thus does away with the slow pacing that has traditionally been the bane of such games.

The game is highly immersive, but does not shy away from humour.

Divinity is still nonetheless very immersive, with plenty of digressing and sidequesting. This is enhanced further with the rich, colourful world, which is second-to-none in the genre. Furthermore, the ability to zoom right into the characters to the point of a third-person perspective is a very nice touch, adding even more immersion.

Website http://store.steampowered.com/app/230230 **Price** £29.99

Even up close, the imgery is stunningly vibrant.

> The story follows the usual fantasy tropes involving dark guilds and plots to doom the world

Saints Row IV

Some crass sandbox silliness.

S*aints Row* goes all out with the sandbox genre, giving the player unbridled freedom in its world, which includes the ability to jump atop skyscrapers and run faster than cars. However, the game's biggest selling point is also one of its biggest weaknesses, since these abilities are available near the beginning of the game, which gives little sense of progression and challenge.

As is expected with the genre, the story is lacklustre, though this is made up for by giving the player plenty to do and through the game's brash humour, silliness and parodies of popular culture. Every aspect, down to the intricate character creation, its dialogue and characters, are designed to have the player in stitches.

In the end, *Saints Row IV* delivers exactly what it promises: a healthy dose of fun. It is for this reason, along with its very attractive price tag, which has made this game so popular.

Website http://store.steampowered.com/app/206420 **Price** £10.99

Saints Row IV doesn't reinvent the wheel, but instead straps fireworks to it.

Grid Autosport

Easily the most realistic racing simulator to be released on Linux.

The car-shaped gap in Linux gaming is gradually being filled, and *Grid* addresses the part of that gap concerned with realistic driving simulation, adding to the battle and arcade racers released recently.

Grid features everything one could want from a racing game: plenty of tracks and cars, realistic physics and satisfying graphics. There's a wide range of classes to choose from, including touring cars and endurance.

The online multplayer can be very enthralling, but a little intimidating to begin with. For the lone wolves, there's a hefty career mode where the player can advance among the different classes and get signed to bigger and better teams as the career progresses. This can be challenging, but for casual players there are options such as the ability to rewind following a crash and display ideal racing lines on the track.

The game features many well known cars and tracks to enjoy.

It's hard to pick any holes in *Grid Autosport*. It's a must have for those more serious about their racing games, while also recommended to those seeking something more casual.

Website http://store.steampowered.com/app/255220 **Price** £24.99

ALSO RELEASED...

Valhalla Hills
This nice little city builder requires the player to expand a Viking settlement up a hill to the point where it reaches Valhalla. Inspired by the likes of *The Settlers*, it incorporates strategy and resource management elements. However, be warned that a few of its mechanics, particularly the pathfinding, tend to be a little on the clunky side and can often be frustrating, though its graphics are certainly relaxing.
http://store.steampowered.com/app/351910

Dreamfall Chapters
This game perhaps deserves more attention, but was plagued with performance issues which have only recently been solved. It's a worthy successor to the *Longest Journey* adventure games, featuring a rich story, memorable characters and extremely intriguing world. Though the puzzles can be a bit hit and miss, the game more than makes up for it with its excellent plot and narrative.
http://store.steampowered.com/app/237850

Vendetta – Curse of Raven's Cry
This game has a troubled history, being released early last year and universally panned for its numerous bugs. This re-release addresses those issues and demonstrates that behind all the issues was a very decent RPG involving pirates, ship-to-ship combat and a satisfying open world. However, it still isn't without its issues and those looking to dabble in a bit of piracy may want to wait for a sale or price drop.
http://store.steampowered.com/app/386280

Beyond legacy code

Ben Everard avoids legacy code problems by using low-quality hard drives.

Author David Scott Bernstein
Publisher Pragmatic Bookshelf
Price £25.50
ISBN 978-1680500790

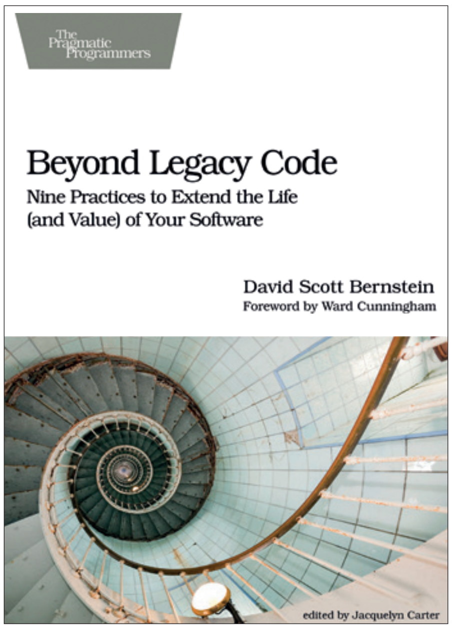
Legacy code refers to the vast number of programs running today that are hard to maintain and prone to breaking. They're a huge source of problems for anyone working in the IT industry, from developers who have to wrestle with them in order to add new features to sysadmins who have to keep old distros running because the legacy code relies on an end-of-lifed package. *Beyond Legacy Code* isn't a book about how to solve the problem with the legacy code we have now, it's about how to solve the problem we will have with legacy code in the future. In other words, it's about writing code that will stand the test of time and remain useful for years.

David Bernstein prescribes an agile approach to this problem. He is keen to point out the importance of actually being agile, not just adding an agile veneer to outdated software development methodologies. His approach is summed up in nine practices that, when properly understood, ensure that not only is agile development used, but its benefits are actually realised.

Bernstein's advice is based on a huge amount of experience, and it's not only easy to follow, but easy to understand. This is important because it's only with this understanding that the principles can be properly applied to each project.

Fewer than half of all software projects are successful. If more people followed the book, that would increase dramatically.

★★★★☆


The Pragmatic Programmers
Beyond Legacy Code
Nine Practices to Extend the Life (and Value) of Your Software
David Scott Bernstein
Foreword by Ward Cunningham
edited by Jacquelyn Carter

Learning to write easy-to-maintain code is a gift to your future self.

Reactive Programming with RxJS

Ben Everard attempts to code using Newton's third law of motion.

Author Sergi Mansilla
Publisher Pragmatic Bookshelf
Price £12.18
ISBN 978-1680501292

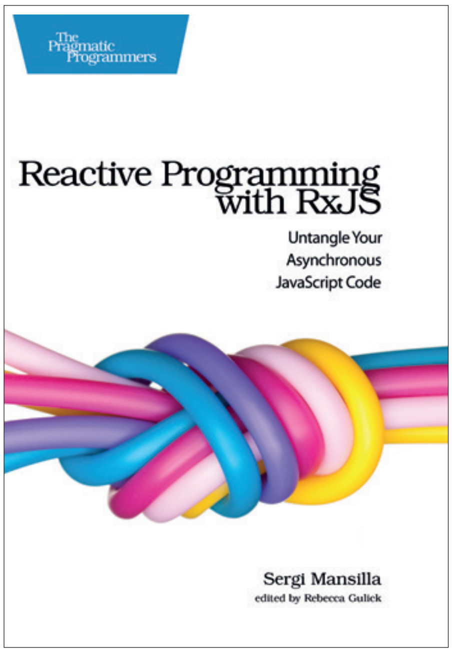
Don't be confused: this is a book about JavaScript with the word Reactive in the title, but it's not a another book about building reactive websites. RxJS is about a the reactive style of programming (this is not unique to JavaScript; it originated in .NET), rather than website design. In this style of programming, data is dealt with in streams that can be filtered and acted upon. These streams (known as observables) are particularly effective for dealing with asynchronous data sources.

Reactive Programming with RxJS focusses specifically on Reactive development rather than on general JavaScript, so the reader has to be familiar with the language, and it will help if they have experience in asynchronous programming. Sergi Mansilla wastes no time in getting down to the nitty gritty of RxJS and how it can create readable, clean code where before lay a mess of callbacks and event listeners.

While Reactive Development exists in most languages, this book is probably a little too tied to JavaScript to be useful to anyone looking to gain the advantages in other languages. However, *Reactive Programming with RxJS* does cover RxJS in Node.js as well as the browser, so with this one tome, you gain the skills to implement the programming style both on the server and for your end users.

A straightforward guide to get you started with RxJS quickly.

★★★½☆

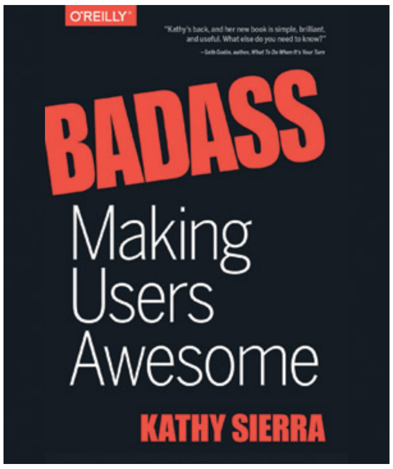

The Pragmatic Programmers
Reactive Programming with RxJS
Untangle Your Asynchronous JavaScript Code
Sergi Mansilla
edited by Rebecca Gulick

RxJS: Code as neat as a knot of rainbow-coloured cables.

Badass: Making Users Awesome

Graham Morrison finally gets another Kathy Sierra book.

Author Kathy Sierra
Publisher O'Reilly
Price £19.99
ISBN 978-1491919019


O'REILLY
"Kathy's back, and her new book is simple, brilliant, and useful. What else do you need to know?"
BADASS
Making Users Awesome
KATHY SIERRA

We've been huge fans of Kathy Sierra for a long time. We first discovered her work through O'Reilly's 'Head First' series of books, which Kathy helped to formulate after co-authoring the first title, *Head First Java*, back in 2003. Written by Kathy and her partner, Bert Bates, it combined her extensive Java training experience (from Sun Microsystems) with her ability maximise attention and concentration, perhaps thanks to her prior experience as a games programmer.

That first book contained many of the elements that have gone on to make the series so useful. But mostly, it was successful because of what it did not contain: in comparison to the average O'Reilly book, there were far, far fewer words. There were no lengthy essays, no wordy diatribes on language syntax, no backgrounds or biography and definitely no reference material. Each book was composed of simple ideas linked together by meme-like images, phrases, charts and remarkable insight. These components were sewn together to engage the reader from the beginning right through to the end.

In theory, tackling big subjects like Java in this way shouldn't work. It's a complex language that can take years to master. But *Head First...* succeeds, not because it's a sugar rush of quick and clever ideas, but because it understands the limitations of teaching a subject through words. The writers probably assume, for example, that most people who start a complex O'Reilly book don't finish a complex O'Reilly book. Yet most of us shouldn't have trouble finishing any *Head First...* title, and even re-reading it several times over. And that's the series' killer feature; accessibility via a 'natural language' style of teaching.

After suffering serious online harassment in 2007, Kathy Sierra stepped out of the limelight, and it has understandably taken her a long time to venture back – first with some excellent blog posts, then with some presentations, and now finally with a new book very much in the *Head First...* style.

"Too often, the goals of a company and the goals of its users aren't just different but mutually exclusive."

Badass: Making Users Awesome is a worthy return, and it's wonderful to see her and her trademark teaching approach back in print. *Badass* doesn't take on the complexity of a language like Java, but it does attempt to answer a vital question relevant to many of us, "Given competing equally-priced, equally- promoted products, why are some products far more successful than others?".

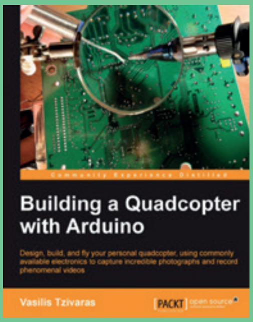
Whether your product is an app, a game, a blog post or a print magazine, *Badass* takes an othogonal approach to helping you understand how best to reach your audience, purely from your potential user's/reader's perspective. It does this by persuading you to forget the 'brand' and concentrate on making your users 'feel' awesome after using your creation. Only then will they become your 'badass' users and help your endeavour to succeed. It's a noble approach that's utterly right.

A fabulous read, although it's unlikely to make us awesome overnight.
★★★★★

Also released... March 2016

Building a Quadcopter

This book promises to go through the nuts and bolts of building a quadcopter based on the Arduino micro-controller. We really like this idea, because quadcopters are becoming very popular, which has made the hardware cheap and accessible. Putting one together yourself or as a project with your kids is a great way of breaking down a complex problem into simple parts.



Build your own open source quadcopter!

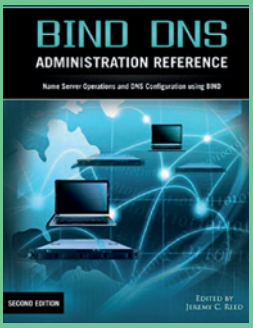
Android Programming

This is a book aimed at beginners, written by an author who we like, with plenty of experience. This is all good news, because mobile operating systems like Android are the new semi-closed PC platform. The more of us writing open source software for it, the better the future of the platform and our hardware. It's also a great way of making you more employable.

Need a new career? Why not try Android coding?

BIND/DNS Admin Reference

This month's sysadmin pages, at the back of the magazine, dive into the complex world Domain Name Servers and the Berkeley Internet Name Daemon, two fascinating and powerful services that hold the web together. There are plenty of books on the subject, for further reading, but this second edition is the latest if you want to take your DNS learning further.



It's difficult, but that makes it special.

GROUP TEST

If you're starting a new Free Software or open source project, you need to pick an appropriate license for it. Here are our recommendations.

On test

GNU GPL

URL www.gnu.org/licenses/gpl-3.0.en.html
Originator Free Software Foundation
The license behind the Linux kernel and other big-name projects.

Affero GPL

URL www.gnu.org/licenses/agpl.html
Originator Free Software Foundation
Like the GNU GPL, but with extra clauses to make sure our freedoms are not curtailed by "software as a service".

BSD Licence

URL www.opensource.org/licenses/BSD-3-Clause
Originator Regents of the University of California
An open source licence allows code to be pulled in to proprietary apps.

Artistic Licence 2.0

URL www.perlfoundation.org/artistic_license_2_0
Originator The Perl Foundation
Written by Larry Wall and used for many Perl implementations and modules.

GNU Lesser GPL

URL www.gnu.org/copyleft/lesser.html
Originator Free Software Foundation
A version of the GPL that lets Free Software libraries be used by proprietary software.

Mozilla Public Licence

URL www.mozilla.org/en-US/MPL/2.0/
Originator Mozilla Foundation
Used by Firefox, Thunderbird, LibreOffice and other apps, aiming to bridge the gap between GPL and BSD.

Free software licences

In the proprietary software world, very few people read software licences. Many of us who occasionally spend time on Windows or Mac OS X are familiar with huge EULAs (End User Licence Agreements) which go on for thousands of words and contain ridiculous amounts of jargon and legalese. So virtually everyone just clicks "I accept" without paying proper attention.

In fact, the situation is so bad that one company, PC Pitstop, put a note in its software's EULA saying: the first person to email a specific address will get $1,000. It took four months and over 3,000 downloads before someone actually read the EULA in full and requested the prize.

Here in the GNU/Linux world, it's very different. Our licences are intended to protect our freedoms, rather than take them away, and they're usually written with real people in mind and not just lawyers. They're shorter, they're clearer, and they try to explain how they want to help developers, users and (sometimes) software vendors benefit from access to source code.

But there are many licences – and the differences can have a big impact on the success of an open source project. So for this month's Group Test, we thought we'd step aside from software and look at the licences behind it. It's good to be aware of the differences as a regular Linux users, but it's even more important if you plan to start an open source project in the future, or at least contribute to one. We'll focus on picking out the ones that are the most beneficial to our software ecosystem as a whole.

There are many licences, and the differences can have a big impact on the success of a project

Free Software vs Open Source

The terms "Free Software" and "open source" may seem interchangeable, but there are some subtle differences in their philosophies.

Free Software is the name used by the Free Software Foundation and GNU project from their early days, and it means "free as in speech, not as in beer". The Free Software Foundation is keen to stress the importance of freedom, community and sharing in its software – social benefits that improve the world. Open source, on the other hand, is a term that was developed in the 1990s to make the concept of Free Software more business friendly. It focuses on the practical aspects of software whose source code you can view, change and distribute: better reliability, more security, and more eyeballs looking at the code. Many people use "FOSS" (Free and Open Source Software) as a term to encapsulate both approaches. You may even come across "FLOSS", where the "L" stands for "libre" – emphasising freedom of speech and not just zero cost.

A single-clause licence

Do What the 'Flip' You Want to Public Licence.

Most of the licences in this Group Test are geared towards large Free Software and open source projects with multiple developers and thousands or millions of lines of source code. But what if you've written some small snippet of code that you want to share with the world, but you don't want the complexity of picking a licence, making sure all your code is compliant with it, and distributing it with your work (especially if it's longer than the code itself)?

Here's where you could use the WTFPL, or Do What the 'Flip' You Want to Public Licence. (You've probably gathered by now that 'Flip' is actually another word in the real licence text...) It's a single-clause licence that has a tiny preamble and then says:

0. You just DO WHAT THE 'FLIP' YOU WANT TO.

In other words, no worrying about rights for distribution, modification, linking, patents or anything like that – developers can simply take the code and do anything with it. Effectively, it's very much the same as putting your code in the public domain. It's not very popular, though, as most open source developers still want some credit for their work, even if they don't want to impose any other restrictions.

GNU GPL

Freedom through copyleft.

The GNU project has contributed so much to what we call 'Linux' today (hence the use of 'GNU/Linux' by some advocates): the GNU C library, *GCC* compiler suite, *Emacs* and many other major software components that make up a free operating system. But equally important is the licence that GNU created: the General Public Licence. This is not just a piece of text describing what you can and cannot do with source code, but a powerful document taking important social and ethical positions, with a goal of preserving and furthering our software freedoms.

Back when Richard Stallman originally developed the GPL, the main other 'open source' licence was the BSD Licence. This was just a few paragraphs saying: here's some source code, do what you want with it, but give the original developers credit (we cover this licence and its impact in detail later). While this licence provided developers with plenty of freedoms, it didn't forcibly preserve those freedoms. Anyone could take BSD-licenced code and tuck it away in proprietary software, so the original developers couldn't benefit from updates.

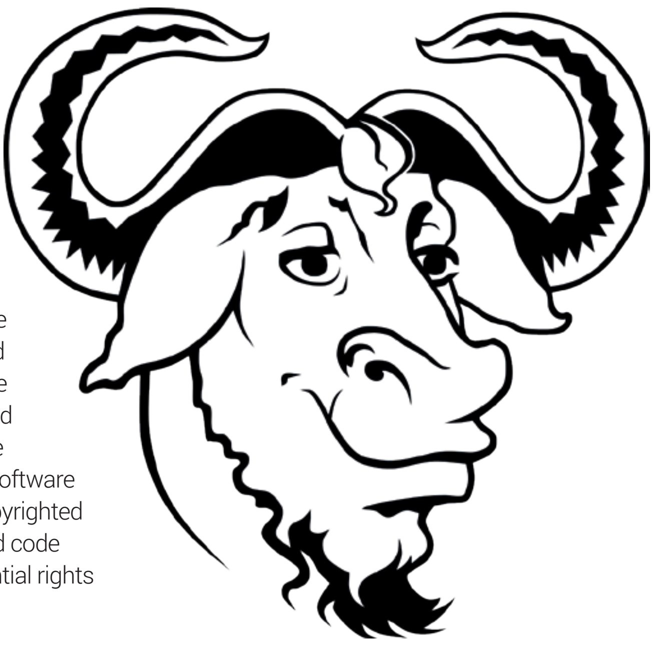
Right from the start, the GPL takes a different approach. One of its opening lines is: "The licences for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public Licence is intended to guarantee your freedom to share and change all versions of a program." This "guarantee" is crucially important: if you release your code under the GPL, it will always remain open for anyone to download, study, modify and redistribute

But what part guarantees these freedoms? Well, Stallman craftily used copyright law – usually intended to prevent people from copying material – and used it to ensure that freedoms can't be taken away. Because GPL-licenced software is copyrighted (eg by the program's author or by the Free Software Foundation), and the GPL is a copyrighted document as well, users of GPLed code cannot omit any of the four essential rights when modifying or distributing it.

The four freedoms

And these four rights are: the right to run the program as you wish; the right to study how it works and modify it; the right to distribute copies; and the right to distribute your modified version. The GPL guarantees that these freedoms are preserved, which is a boon for most of us, but can cause issues for some developers working on projects containing a mixture of open source and proprietary code.

Indeed, many proprietary companies have put into place a "no GPL" policy – don't even look at or touch any GPLed code, because if it somehow ends up in our product, we may have to release the whole thing under the GPL. We don't find this a big issue, however – the GPL is all about our freedoms as users and hackers, and not to make life easier for megacorps.


The GNU Project has brought us many awesome components for a free OS, but also a philosophy and licences to match.

Ultimately, the GPL is a rather long document at 5,645 words (although still far shorter than many proprietary app EULAs), and some developers take issue with its political stance and "enforcement" (rather than optional offering) of freedoms. But we think it's an incredibly well thought-out document that has made the FOSS community so strong, and just keeps getting more useful with each year.

VERDICT
The GPL embodies the principles of sharing and community that are so vital to Free Software.
★★★★★

Artistic Licence

As used by Perl and many modules.

Perl may look rather old-school today when everyone is raving about hot and trendy languages such as Go and Rust, but it's still an important language that's doing plenty of work on servers around the world – especially in text processing jobs. Some coders criticise Perl for looking like line noise, whereas others praise how much functionality the language can pack into just a few characters.

Perl is released under the Artistic Licence, a curious text that has gone through some major modifications over the years. The original 1.0 version received a lot of criticism for being too vague in places; Bradley Kuhn (as interviewed last issue) worked with the Perl team and Free Software Foundation to create the Clarified Artistic Licence, which is the current one used by Perl. Future versions of Perl will use Artistic Licence 2.0, and some other apps such as the *SNEsE* Super Nintendo emulator already use this newer version.

So, what does the licence specify? Well, it's very heavily focused on what you can do with modified versions of a program. You're allowed to take some code released under the Artistic Licence and make another program out of it, providing that you "clearly document how it differs from the standard version". In other words, you demonstrate what you've changed, what you've added, what features you've implemented and so forth.

You must also ensure that users can run both the original version of the software and your modified version simultaneously. So if you decide to fork Perl, for instance, you need to make sure that your new version doesn't trample all over the official release (and Perl programs on a freshly installed Linux installation don't run your version by default).

It's focused on what you can do with modified versions of a program

Perl 6 will use version 2.0 of the Artistic Licence. Here's the language's cheerful mascot, Camelia.

VERDICT
Worth using if your code is likely to be modified or forked in future.
★★★☆☆

Mozilla Public Licence

As used by Firefox and LibreOffice.

As we've seen, the GPL is the best choice of license if your priority is freedom for end users – that is, the freedom to study and modify all parts of your software, and these freedoms cannot be taken away. The BSD Licence, in contrast, focuses more on freedom for developers; specifically, the freedom to use code in proprietary products if desired. Both of these licences serve their target groups well, but is there a middle point?

The Mozilla Public Licence may be the solution here. This was originally written by Mitchell Baker of Netscape back in the late 90s, as the browser developer was toying with the idea of going open source. In its first incarnation as the Netscape Public Licence, it allowed code developed in the open source community to be incorporated into proprietary products, which didn't win applause from Free Software developers. So the Mozilla Public Licence (MPL) was born, and it takes a clever approach: code that is licensed under the MPL must remain under the MPL, even after modification, so it always remains free like in GPLed code. But! It is possible to mix MPLed code with proprietary code, creating a proprietary product. If you make such a product, you don't have to release the proprietary code you wrote yourself, but you must make available the MPL code you used along with any changes to it.

Caudine fork

In this way, commercial developers can use MPLed code such as the source tree for *Firefox*, *LibreOffice* and other flagship open source apps, and build products on top of it. If they add their own separate features, they don't have to release the code for those, and they get a commercial benefit over the competition. But if they modify MPLed code created by the community, we must get the changes back to benefit us too.

Want to keep your code open, but don't mind it being combined with proprietary code? Mozilla has the solution.

VERDICT
A healthy compromise between Free Software and proprietary.
★★★★☆

BSD Licence

Permissive and proprietary friendly.

The FreeBSD operating system is the most prominent example of a BSD Licensed project.

Like the GNU GPL, this licence was originally drafted with a specific operating system in mind, although today it is used by thousands of projects. BSD refers to the Berkeley Software Distribution, a flavour of Unix that was developed at the University of California, Berkeley, from 1977 to the mid 90s. Although BSD itself is no longer developed, a handful of forks developed from it in the early 90s, most notably FreeBSD, OpenBSD and NetBSD (famous for being portable to almost anything).

As it stands today, the licence has three requirements. You're allowed to redistribute the software in source and binary formats, with or without modification, providing that:

1. The source code contains a copyright notice and disclaimer (the latter saying that the software is provided as-is, and with no warranty).
2. If distributed in binary only (ie proprietary software), the same copyright notice and disclaimer is included in the documentation.
3. The names of the original developers cannot be used to endorse or promote software derived from it without specific written permission.

So in short: do what you want with it, give us credit for writing it, don't sue us if it breaks, and don't claim we support your spin-off. Note that unlike the GNU GPL, there is no clause here saying that source code must be supplied. It's a permissive licence – there are very few requirements compared to the GPL.

Who wins in the end?

Now, the side effect of this licence is clear: any company can take BSD-licensed code and incorporate it in a proprietary product. This has happened many times, such as when Microsoft built a networking stack for Windows using BSD code, or more recently with the PlayStation 4, which runs a modified version of FreeBSD. For some people, this makes the BSD licence severely flawed – after all, why should we help companies that often don't provide a penny or single line of code back?

Well, it depends on your perspective. The BSD Licence is arguably better than the GNU GPL in that it gives other developers increased freedom and choice (although at the expense of user freedoms). So if you're writing some code and simply want to get it out there and used as much as possible, without any philosophy or politics, it makes sense.

VERDICT
A minimal, simple choice if you just want to get your code out there.
★★★★☆

Another perspective on BSD

How the licence encourages adoption.

One argument in favour of BSD-style permissive licences that often comes up is adoption of key technology and standards. If you want to get the whole world using a certain protocol, library or piece of software, you want to use the licence that's palatable to as many people as possible.

One example of this is OpenSSH, the secure remote shell tool ubiquitous in every modern-ish operating system (even Microsoft is starting to use it). BSD fans have often claimed that the permissive licence has helped with its adoption, because any company can start using it without having to think of legal issues or releasing changes back to the world.

OpenSSH is a well tested and engineered piece of software, and thanks to the BSD licence, very few people have a beef using it. Had it been released under the GPL, a lot of companies and software vendors could have had concerns using it, and so created their own versions and we'd have lots of potentially insecure, slightly incompatible versions all over the internet. Maybe some companies would have forked older BSD Licensed versions and the situation would be a jolly big mess.

But as it is, OpenSSH has become the *de facto* standard for command line access to remote machines, and with pretty much the whole world using (and examining) the same codebase, we have a strong project with a good security track record and no major forks.

The BSD Licence has helped one open source project to utterly dominate in a certain task.

GNU Lesser GPL vs Affero GPL

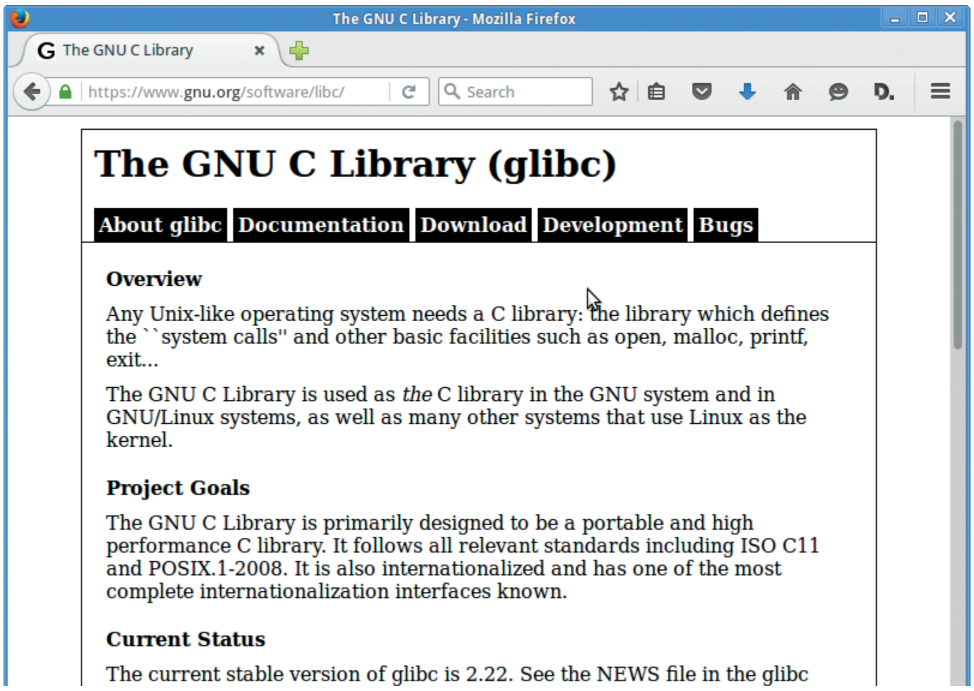
What do these two derivatives offer?

Richard Stallman and the Free Software Foundation often get flak for not budging one inch when it comes to other philosophies and ideas. But we think this is unfair, and one example of when the FSF has been able to compromise is the GNU Lesser GPL (aka LGPL).

Essentially, the LGPL is very much identical to the regular GPL, but includes an exception: it lets proprietary software link to LGPLed software so that the former can use the facilities of the latter. This is especially important in terms of libraries – collections of software routines that an end-user doesn't run on their own, but which provide support for other programs.

For instance, the GNU C Library (*Glibc*), which provides various input, maths and other routines, is released under the LGPL. So a proprietary program can link to it and use its routines, without that program having to be Free Software (released under the GPL) as well. Many other libraries have been made available under the LGPL, so it's not uncommon to have a piece of proprietary software that spends most of its execution time inside routines provided by LGPLed libraries.

Now, given that the goal of the GNU project is to have a completely open source operating system, why cater to proprietary app developers? Well, when a Free Software library doesn't offer many more features than a proprietary one, it's the lesser of two evils if a proprietary app uses the free library.


The GNU C Library - Mozilla Firefox
The GNU C Library
https://www.gnu.org/software/libc/
Search

The GNU C Library (glibc)

About glibc | Documentation | Download | Development | Bugs

Overview

Any Unix-like operating system needs a C library: the library which defines the ``system calls'' and other basic facilities such as open, malloc, printf, exit...

The GNU C Library is used as *the* C library in the GNU system and in GNU/Linux systems, as well as many other systems that use Linux as the kernel.

Project Goals

The GNU C Library is primarily designed to be a portable and high performance C library. It follows all relevant standards including ISO C11 and POSIX.1-2008. It is also internationalized and has one of the most complete internationalization interfaces known.

Current Status

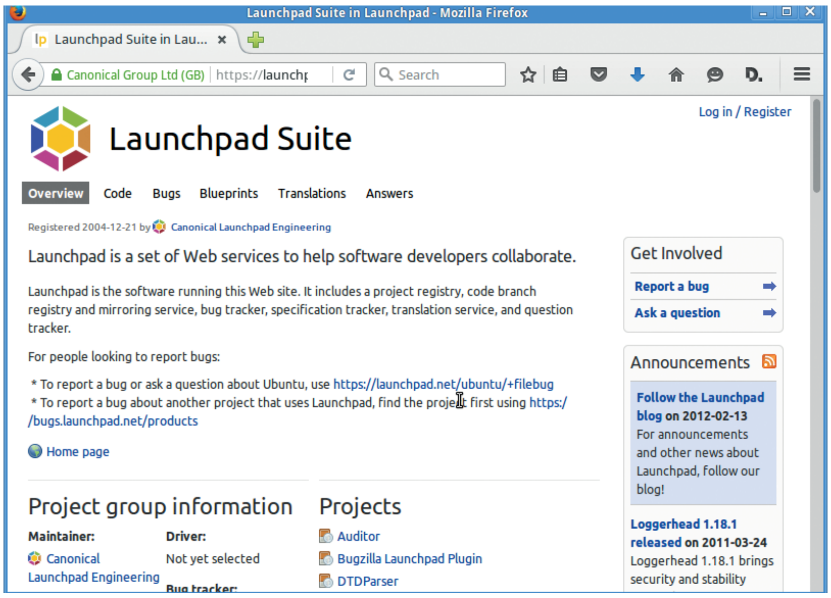
The current stable version of glibc is 2.22. See the NEWS file in the glibc

The GNU C Library is an important component of the GNU/Linux system, and is released under the LGPL.

The Affero alternative

The Affero licence tackles a different problem. If someone gives you the binary of a GPLed program to run on your computer, they must (on request) give you the source code and licence as well. But what if you're not running GPLed software directly, but over the internet? Think of all the web apps out there, like web-based email clients: you're not technically running the software on your machine, but merely viewing the results of it running on a remote one, so should you then be able to request access to the source if that code is GPLed?

Long before the emergence of cloud computing, which brings up this very issue, Richard Stallman and others had noticed that it could become a problem. So work began on the Affero Public Licence (aka AGPL), which is very similar to the standard GPL but includes an extra clause dealing with "remote network interaction" and says: if people interact with your AGPLed software over a network, they have the right to request the source code to that software – and at no extra charge.


Launchpad Suite in Launchpad - Mozilla Firefox
Launchpad Suite in Lau...
Canonical Group Ltd (GB) https://launchp
Search
Log in / Register
Overview
Code
Bugs
Blueprints
Translations
Answers

Launchpad Suite

Registered 2004-12-21 by Canonical Launchpad Engineering

Launchpad is a set of Web services to help software developers collaborate.

Launchpad is the software running this Web site. It includes a project registry, code branch registry and mirroring service, bug tracker, specification tracker, translation service, and question tracker.

For people looking to report bugs:

* To report a bug or ask a question about Ubuntu, use https://launchpad.net/ubuntu/+filebug
* To report a bug about another project that uses Launchpad, find the project first using https://bugs.launchpad.net/products

Home page

Get Involved
Report a bug
Ask a question

Announcements

Follow the Launchpad blog on 2012-02-13
For announcements and other news about Launchpad, follow our blog!

Loggerhead 1.18.1 released on 2011-03-24
Loggerhead 1.18.1 brings security and stability

Project group information

Maintainer: Canonical Launchpad Engineering
Driver: Not yet selected
Bug tracker:

Projects
Auditor
Bugzilla Launchpad Plugin
DTDParser

Launchpad, Canonical's service to help developers collaborate, is released under the AGPL.

VERDICT

LGPL Makes some compromises to proprietary developers for the greater good.
★★★☆☆

AFFERO A simple and effective solution to the use of GPLed software in web apps.
★★★★☆

OUR VERDICT

Free software licences

Which license is intrinsically 'best' is very much a matter of opinion, and we know that some readers may disagree with our findings here. And ultimately, different types of project have different licensing requirements, so we won't say that one is simply worth using over another in every single possible circumstance. What we at Linux Voice can say, however, is that the GNU GPL is the one that has really helped us to get where we are today.

Yes, it's wordy. Yes, it's political. And yes, it causes consternation for some software developers who also work on proprietary products. But it's not some crusty old text that we still use out of habit: it's a very well thought-out document that deals with today's concerns despite originally being written decades ago. Richard Stallman foresaw how the software world would develop, and created a licence to protect his work against it.

Without a powerful, copyleft, freedom-centric licence like the GPL, the world would look very different today. Sure, we'd still have some open source projects like the BSDs, but thanks to the GPL we have an enormous tapestry of source code out there that was, is, and always will be free. The GPL makes us think about the social benefits of Free Software, not just the practical ones, so for that it should be praised highly.

Of course, the other licences have their benefits as well. Although we regard the GPL as the ultimate licence for our freedoms, the BSD Licence is still a great choice when you simply want your code to be used anywhere, regardless of whether the end result is open or closed. The BSD Licence is clear, simple and short, and in many cases companies that use BSD code still contribute their changes back (such as LLVM/Clang).

Artistic, Mozilla, LGPL and AGPL all have their goals and purposes too. But ultimately, the original GPL embodies the spirit and community of Free Software to the largest extent, and that's why we love it.

> Without a powerful, freedom-centric licence like the GPL, the world would look very different today

The MIT Licence

Some readers may be wondering why we haven't included the MIT Licence here, given that it has become one of the most popular for open source development in the last few years. Well, we've omitted it for the simple reason that it's almost the same as the BSD Licence, but with one clause removed. Like the BSD Licence, the MIT Licence says you can do what you want with the software, there's no warranty provided with it, and you must include a copyright notice and the licence with the software.

However, the MIT Licence omits the clause stating that the names of the original developers cannot be used to endorse software based on it. So it's a slightly simpler version of the BSD Licence, and is otherwise the same. Is it a big deal? Well, in most cases, no. If you have a popular piece of MIT Licenced source code and someone makes a fork that's utterly rubbish but says you're championing it, very few people will believe that person.

If you're working on something new, however, and don't want lots of half-quality forks cropping up with people using your name to promote them, the BSD Licence is a more sensible choice.

The GNU General Public License does the most to preserve and promote our freedoms in the long run.

1st GNU GPL

www.gnu.org/licenses/gpl-3.0.en.html
A cornerstone of the modern Free Software movement, and a license that puts users' freedoms at the forefront.

2nd BSD Licence

www.opensource.org/licenses/BSD-3-Clause
Has very few requirements and permits use in proprietary products. But as we've seen, that's not always bad.

3rd Mozilla Public Licence

www.mozilla.org/en-US/MPL/2.0
A clever licence that helps to bridge the gaps between two approaches to software development.

4th Affero GPL

www.gnu.org/licenses/agpl.html
An important fork of the GPL that deals with something we're seeing more of: apps running in our browsers.

5th GNU Lesser GPL

www.gnu.org/copyleft/lesser.html
GPLed software is the best, we feel, but sometimes an LGPLed library is the lesser of two evils.

6th Artistic Licence

www.perlfoundation.org/artistic_license_2_0
This licence has worked well for Perl and many of its modules over the last few decades.

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 100 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 LINUXVOICE

ON SALE THURSDAY 25 FEBRUARY

Image: iStock

HACK EVERYTHING

The hackers don't stand still, and neither should you – learn their tricks to keep yourself safe online.

EVEN MORE AWESOME!

SASS

Like the idea of CSS (and the attractive, consistent web layouts it produces) but hate the arbitrary fiddliness of it? We have the tool for you and your website.

Media players

Find the tools Free Software has to offer with which to depress yourself watching *Truly Madly Deeply* and console yourself listening to side 1 of *Low*.

Fosdem

Mike Saunders braves waffles, beer and moules-frites to report on Europe's biggest FLOSS gathering. Will he come back in one piece? Find out!

LINUX VOICE IS BROUGHT TO YOU BY

Editor Graham Morrison
graham@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Technical editor Ben Everard
ben@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

Copyright Linux is a trademark of Linus Torvalds, and is used with permission. Anything in this magazine may not be reproduced without permission of the editor, until September 2016 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2016
ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**

FOSSpicks

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

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

Circuit simulator

Hardware designer

More of us than ever before are now messing around with electronics, whether that's building a 3D printer for your local hackspace or creating your own circuits for short production runs. But unlike learning to program, where mistakes are a part of the learning process, the consequences for mistakes in circuits can be more serious.

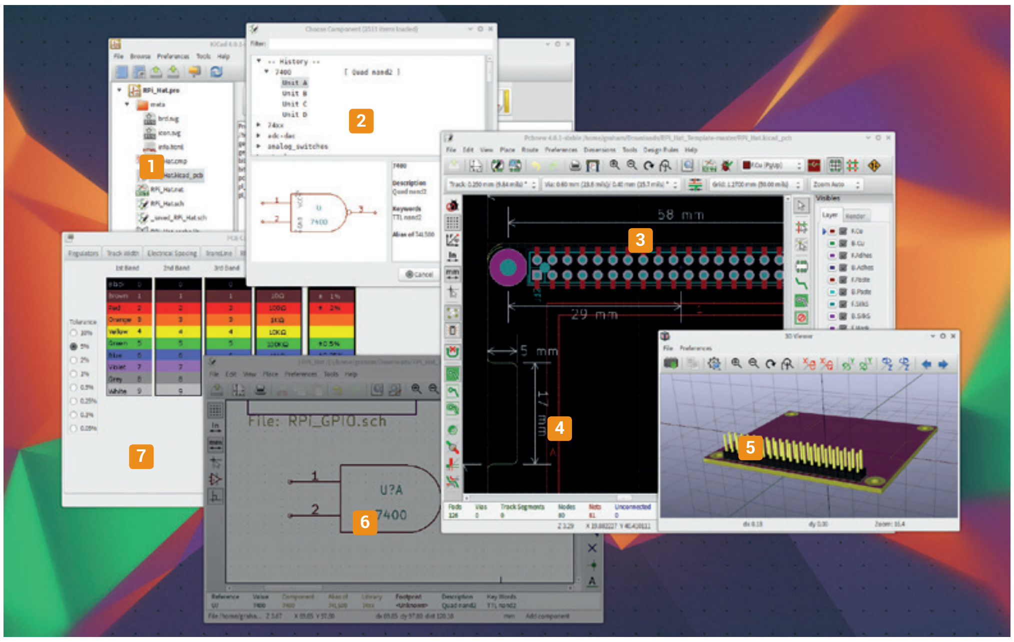
Circuit simulators (or Electronic Design Automation suites) can help with this, not only by error checking your designs, but by understanding the properties of the components you're using. They're roughly analogous to IDEs for programmers, with automated layout and component libraries replacing syntax highlighting and API reference material.

Better by design

KiCad is one of the most comprehensive EAD suits we've found. It's got a long history – the initial release was in 1992 – and recent development has been partly funded by CERN.

It's capable of serious results too, taking you from schematics design to PCB layout, and while it will struggle with large projects it's more than capable of handling homegrown projects. There's even a 3D viewer, which is perfect for studying other designs or visualising your own prototypes. The only disadvantage is that these capabilities come with complexity, and it's a difficult suite to get started with. The schematic designer is very powerful, for example, letting you drag and drop components into your designs and change their values easily, but you still need a good grounding in electronics for those circuits to make sense, even with the Electrical Rules Checker, which checks for logic errors against component and pin types.

Even if you are a beginner, *KiCad* is a worthwhile installation, and this being open source, many designers have shared their own schematics which can be altered relatively easily. Version 4.0 is a major update and includes OpenGL accelerated *Cairo*-enabled rendering, new libraries and the excellent 3D board rendering, which looks brilliant on complex circuits. There's also CERN's interactive Push and Shove router, which has been part of the alpha/beta since 2013. This is for advanced users, and way above our electronics ability, but it promises automatically 'pushed' track routing when creating your own designs, which at least looks really impressive in the YouTube videos explaining its function.



1 **Project Overview** Each project contains lots of different elements, from the schematics to a Bill of Materials
2 **Library** You can install additional component libraries that list capability and common part numbers
3 **Schematics Designer** Drag and drop components and join them together to make circuits, then test them using the ERC
4 **Router** The push and shove router can make sure your connections are placed as efficiently as possibly
5 **3D View** Visualise your project, including all your components, in lovely 3D OpenGL 6 **Circuit Diagrams** Dropped components from the library can be connected to each other, or imported from other libraries
7 **PCB Calculator** Great for quick reference and for checking and calculating the values of any components you need

PROJECT WEBSITE
http://kicad-pcb.org

Latex editor

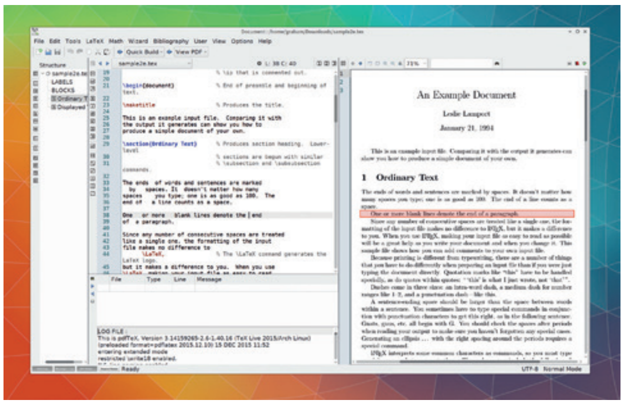
Texmaker 4.5

Latex is a brilliant tool for styling documents, despite sometimes feeling more like programming than authorship. It's capable of stunning results, and unlike the output from a word processor, layouts are dynamic, responsive and adaptable, especially when citing or cross-referencing other documents.

While some purists write *Latex* from *Emacs* and *Vim*, there are several great graphical applications that make things easier, with *Texmaker* being our current favourite. It doesn't exactly make *Latex* easy – you'll still need to hunt down and install your own packages and dependencies, and the real time previews within the application depend on a working configuration outside of the application. We spent some time making sure the command line tool **xelatex** generated predictable output from our *Latex* files first, for example. This is one of the configuration paths that *Texmaker* can use to generate its previews. But with everything installed, creating documents is as a simple as editing HTML.

The preview window defaults to the right, while the text editor lives on the left. Clicking 'Quick Build' generates a PDF by default, which is then previewed. For most documents this is almost instantaneous, and *Texmaker* will highlight any problems it finds, such as text taking up too much space or syntax errors in your document. The vast majority of markup options are easily accessible, with the most common embedded within the left-hand border of the editor itself, and many more are listed in the border of the structure pane. This really helps if you're just beginning with *Latex*, as you don't need to worry about memorising syntax or using two windows for writing.

Creating documents in Texmaker is as simple as editing HTML



```
\begin{document}          % End of preamble and beginning of
text.

\maketitle                % Produces the title.

This is an example input file.  Comparing it with
the output it generates can show you how to
produce a simple document of your own.

\section{Ordinary Text}   % Produces section heading.  Lower-
level
                          % sections are begun with similar
                          % \subsection and \subsubsection
commands.

The ends  of words and sentences are marked
  by   spaces. It  doesn't matter how many
spaces    you type; one is as good as 100.  The
end of   a line counts as a space.

One   or more   blank lines denote the end
of  a paragraph.

Since any number of consecutive spaces are treated
like a single one, the formatting of the input
file makes no difference to
      \LaTeX,             % The \LaTeX command generates the
LaTeX logo.
but it makes a difference to you.  When you use
```

```
This is pdfTeX, Version 3.14159265-2.6-1.40.16 (TeX Live 2015/Arch Linux)
(preloaded format=pdflatex 2015.12.10) 15 DEC 2015 11:52
entering extended mode
restricted \write18 enabled.
```

An Example Document

Leslie Lamport

January 21, 1994

1 Ordinary Text

The package for Arch Linux is the only one to have made the leap from *Qt 4* to *Qt 5*, although you can build your own from source if you want to.

PROJECT WEBSITE
www.xm1math.net/texmaker

Screen capture

Spectacle

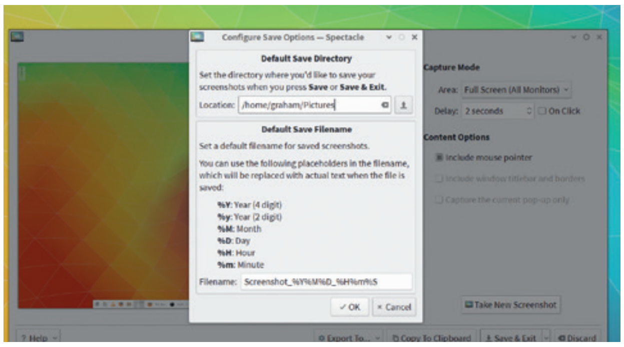
There's been little recent innovation in screen capture tools. This might seem a little unfair considering their basic and utilitarian function, but we feel there's still plenty of potential for new features, especially as we typically take dozens of grabs a week.

Our screengrabber of choice, at least for the KDE desktop, is the default *KSnapshot*. It does everything we need quickly and efficiently – it's triggered by a hotkey, enables you to grab the contents of the entire screen, a window or a draggable rectangle, and saves consecutive images without interaction and with sensible file names.

But *KSnapshot*'s days appear to be over. A recent KDE 5 update has swapped it with a tool called *Spectacle*. *Spectacle* has been freshly developed to take advantage of the new KDE 5 frameworks, and was originally called *KScreenGenie*. *Spectacle*'s developer, Boudhayan Gupta, had intended to simply clean up *KSnapshot* for the new KDE versions, but thought it better to start from scratch after finding bits of code as old as KDE 3.5 in its source tree.

Taking pictures

Despite the complete overhaul, it's almost functionally identical to *KSnapshot*. The only differences we can spot are a neater GUI and more naming options for automatically saved screenshots. It's potentially capable of taking shots of other desktop back-ends, such as Wayland, though support for this hasn't been added yet. This isn't such a problem while Wayland remains in beta, but will become important as more KDE users switch over to the already functional technology preview supported by the last few KDE releases. At least *Spectacle* has this potential, whereas *KSnapshot* does not.



Default Save Directory

Set the directory where you'd like to save your screenshots when you press **Save** or **Save & Exit.**

Location: /home/graham/Pictures

Default Save Filename

Set a default filename for saved screenshots.

You can use the following placeholders in the filename, which will be replaced with actual text when the file is saved:

%Y: Year (4 digit)
%y: Year (2 digit)
%M: Month
%D: Day
%H: Hour
%m: Minute

Filename: Screenshot_%Y%M%D_%H%m%S

Area: Full Screen (All Monitors)
Delay: 2 seconds ☐ On Click
[✓ Include mouse pointer]
☐ Include window titlebar and borders
☐ Capture the current pop-up only

The only problem we have with replacing *KSnapshot* with *Spectacle* is remembering *Spectacle*'s name.

PROJECT WEBSITE
https://github.com/KDE/spectacle

Command trigger

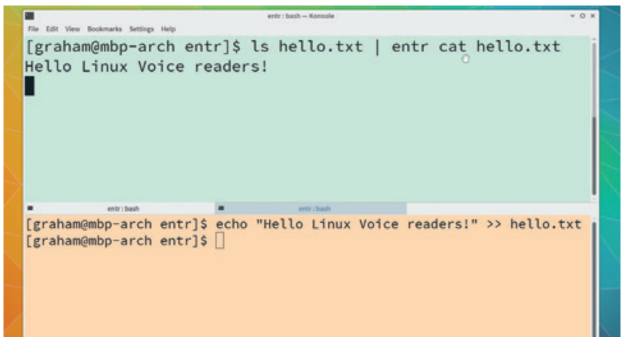
entr(1)

Event Notify Test Runner – *entr* – is one of those command line tools that sticks to tradition – doing one thing well so that lots of other tools can be augmented with its functionality. The old-school charm even applies to the project name, where the developer has included the '(1)' to show where the man page documentation should be found on your system, and subsequently, what category of tool you're using. (1) is for general commands such as this, for example, while (6) is for games like *Nethack* and (8) for system administration. It's a quick way of knowing what kind of tool you're looking at, and *entr* definitely fits into the general category. When you understand what it can do, there's no limit to its possible uses.

Put at its simplest, *entr* enables you to run a command when it detects changes in the filesystem. You could, for instance, force a document viewer to reload when a file is updated, or rebuild a project from source when the code is modified. Or trigger an automatic backup. Or copy files to a remote server. Or transcode a new video or audio download from one format to another. Or shut down your system after a file appears.

What's more impressive is that it's really easy to use. Typing **ls hello.txt | entr cat hello.txt** will ask *entr* to watch the piped output from the **ls** command, and when a change is detected, run the following command. In our example this is **cat hello.txt**. This simple construction will display the updated contents of the **hello.txt** file whenever the file is changed. Additional arguments include **-d** for watching a directory of files, **-p** to avoid running the output command until a file is modified, **-c** to clear the display before running the output and **-r** to reload a persistent child process. With these simple additions, you can automate almost any task with a single command.

entr enables you to run a command when it detects changes in a filesystem



```
[graham@mbp-arch entr]$ ls hello.txt | entr cat hello.txt
Hello Linux Voice readers!
```

```
[graham@mbp-arch entr]$ echo "Hello Linux Voice readers!" >> hello.txt
[graham@mbp-arch entr]$
```

entr will watch for changes or new files in a folder and execute a command when a change is detected.

PROJECT WEBSITE
http://entrproject.org

Compiler

Free Pascal Compiler 3

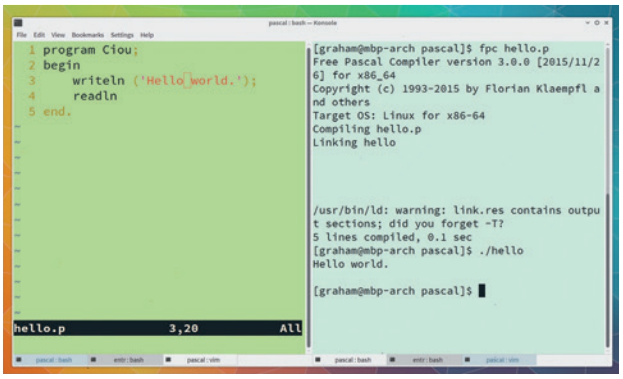
Retro gaming is big business. There are online stores built specifically to sell old games, and re-issues and remasters of classic titles seem to be released every week. The same is also true of old computers, operating systems and languages. Or perhaps, more accurately, the decline of old programming languages just hasn't happened.

Lots of us still play with old languages for a variety of reasons. Our recent series on x86 assembler has been incredibly popular, for example, despite assembler having limited modern uses.

Pascal is another language worth reminiscing over. It's still used in lots of places, but most of us remember it fondly as Turbo Pascal. It was used in many universities to teach the fundamentals of programming. Like BASIC's revival, it's just as good at teaching those procedural concepts today as it was in the 1980s. Which is where this major update to the *Free Pascal Compiler* comes in. Unlike Turbo Pascal, it's just a compiler. There's no IDE or editor for working with Pascal, but it will turn your source code into an executable binary. And it can build binaries on almost any platform, from the Raspberry Pi to the Nintendo Gameboy.

Learn from the past

Free Pascal Compiler also comes with a wonderful set of documentation, and while programming in Pascal does indeed bring us a sense of nostalgia, it's also educational, especially if you've spent the last decade chasing after languages like JavaScript, C# and Swift. Version 3 of *FPC*, as it's known and used on the command line, is a big update that has excellent compatibility with Turbo Pascal 7, and while we've joked about Pascal's age, we know it's still being used in many serious applications and projects, which really gives you the perfect excuse to revisit this gem of a language.



```
program Ciou;
begin
    writeln ('Hello world.');
    readln
end.
```

```
hello.p                    3,20              All
```

```
[graham@mbp-arch pascal]$ fpc hello.p
Free Pascal Compiler version 3.0.0 [2015/11/2
6] for x86_64
Copyright (c) 1993-2015 by Florian Klaempfl a
nd others
Target OS: Linux for x86-64
Compiling hello.p
Linking hello



/usr/bin/ld: warning: link.res contains outpu
t sections; did you forget -T?
5 lines compiled, 0.1 sec
[graham@mbp-arch pascal]$ ./hello
Hello world.

[graham@mbp-arch pascal]$
```

Relive those magical days where your code needed to be compiled and linked before being executed.

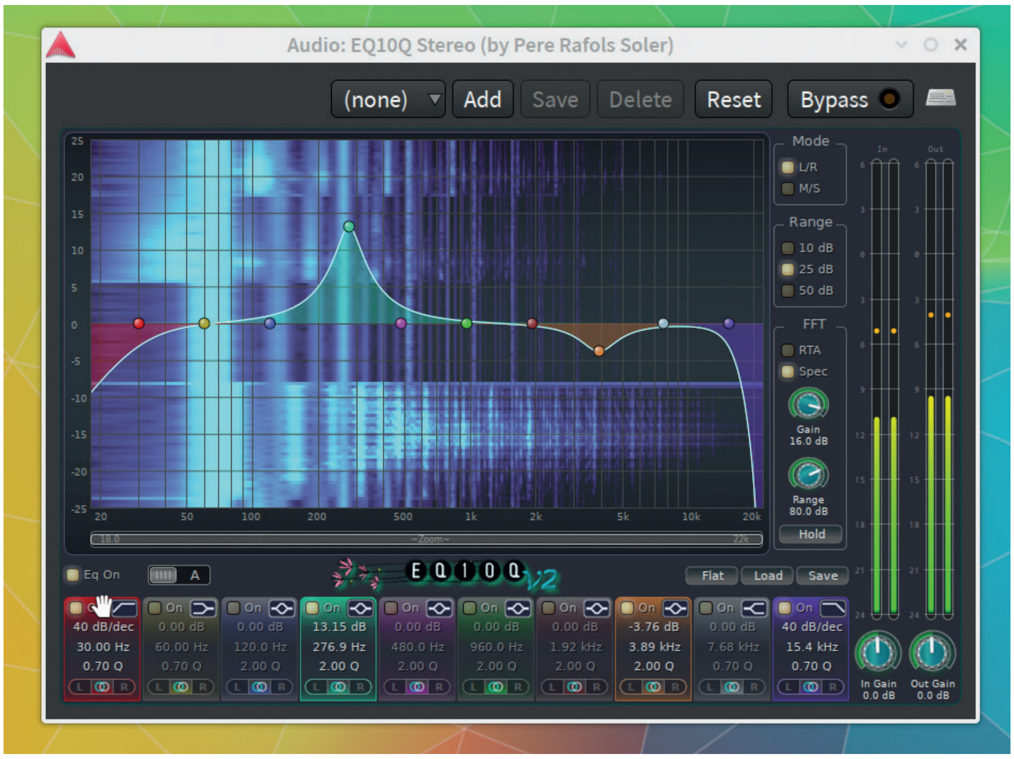
PROJECT WEBSITE
www.freepascal.org

Audio effects

EQ10Q Plugin Pack v2

When listening to music or working with audio, most of us will adjust the EQ of the audio to better suit our tastes and listening environment, even if that's just the dub crowd ramping up the bass for their speaker stack. But outside of boom boxes and Hi-Fis, equalisation is a serious tool that's used both creatively and practically. It's the audio equivalent of adjusting the colour saturation and contrast in an image.

Adding bass and high frequency boost is the audio equivalent of making the black and white elements of an image more pronounced, for example. As such, EQ is an absolutely essential step in any audio production. It's why there's EQ on each of the 24 inputs found on the EMI TG12345, the first solid-state mixing console and the one used to mix both The Beatles' *Abbey Road* and Pink Floyd's *The Dark Side of the Moon*.



Any LV2 compatible host, such as *Audacity* or *Ardour*, can be used to pipe audio through the wonderful effects in this bundle.

The Equaliser

Equalisation is basically filtering. You highlight a range of frequencies in the audio spectrum and choose to either boost or lessen these frequencies.

Old hardware was cherished for the character that this process exerted on the audio in much the same way that a synthesizer's filters define its sound, and modern digital audio engineers have similar preferences for the computational algorithms doing the same thing in software. This is why there's such a huge difference in sound between the practical qualities of *Audacity*'s default EQ, and the sound you get from *EQ10Q*, a spectacular open source EQ LV2 plugin that's part of a small bundle with the same emphasis on sound and control.

EQ10Q can genuinely rival expensive alternatives, not only in its sound characteristics but also in the incredible amount of control it offers. There are different instances for modifying 1, 4, 6 and 10 frequency ranges – you might need a single filter to cut out a 50Hz buzz, for example, or roll off frequencies higher than 18kHz, whereas 10 will let you sculpt the sound into whatever shape you need. Each filter can be enabled and disabled, dragged across any frequency and boosted/diminished with a drag up or down.

But there's a lot more on offer too. You can enable a spectral view of the audio, or a real-time frequency plot, both of which respond to your changes. These help you easily spot hums and buzzes, for example, and work just like colour curves in *Gimp*. Each filter can be switched between one of six different types – low-pass for high frequencies, high-pass for low, and peaks, notches and shelves for those in between. You can quickly switch between two alternate configurations with A/B.

EQ10Q rivals alternatives for its sound characteristics and the control it offers

The equaliser is the main component in a small package of effects that also includes a brilliant compressor and an absolutely essential noise gate. The compressor will make the quieter parts of audio louder using a curve to describe which amplitudes are curtailed and by how much. The noise gate is best used on a noisy mic or guitar input, as it will mute the audio when the level falls below a certain threshold.

All three filters are vital for any audio work, especially with an application like *Ardour* that doesn't include any if its own effects at all – the gate, for example, is absolutely necessary for podcasts and yet there's no other open source gate we've been able to find for our own recordings. As a result, this is a brilliant package.

PROJECT WEBSITE
http://eq10q.sourceforge.net

Digital television recorder

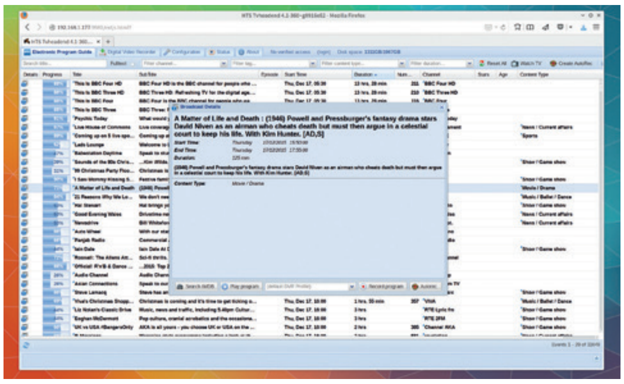
TVHeadend 4.1

We've been using *TVHeadend* for a couple of years. For us, it is without doubt the best system for recording digital television, whether that's terrestrial, cable or satellite. Its best feature is a low resource overhead. *TVHeadend* takes up very little RAM and CPU and can easily be installed on a Raspberry Pi or even a humble NAS.

You start off by making sure your TV-grabbing hardware is automatically detected by Linux. With *TVHeadend* installed and running, you'll then be able to select this as a DVB input though its web interface. After this, create a network and add a 'mux' that connects to the network and the device itself. Muxes are bundles of digitally transmitted channels, and you may need specific information on their formatting, such as the frequency, polarity and FEC for satellite transmissions. This data is all available online, and *TVHeadend* is good at scanning automatically. After a scan, your mux should yield services that can be mapped to actual channels that should then begin to populate themselves with programme data. If not, you'll need to install and configure *XMLTV* too.

After the setup...

You'll be able to browse the EPG, record individual programs or an entire series, or even set up a search term to schedule recordings automatically. Profiles can be used to process these files differently, or save them in different locations – we use one for movies, for example. You can watch recordings or live transmissions from a browser and create accounts for people you'd like to share access with. We use ours in combination with a *TVHeadend* PVR plugin for *Kodi*, allowing us to hide the recording equipment away from the Raspberry Pi connected to the screen, and it works as if both were local to one another. There's also a great Android app.



Whether you want to record a TV series or stream live to *VLC* and *Kodi*, *TVHeadend* is the best television recorder we've ever used.

PROJECT WEBSITE
https://tvheadend.org

Image viewer

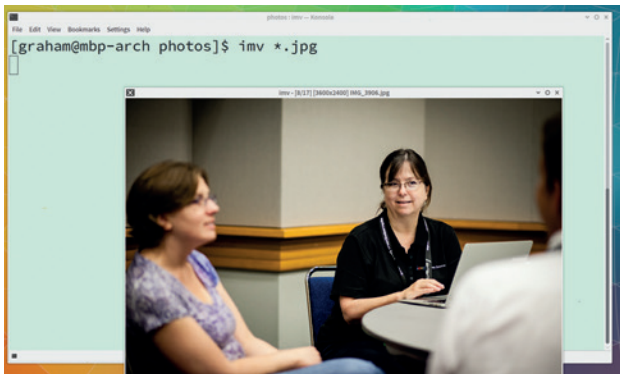
imv 1.2

There have been many image viewers. We fondly remember *Xv* on the Amiga, and the way the same program could be found in Mandrake Linux, helping with the transition from one system to another. This is where *imv* is going to help too. Not only is this a rather nifty image viewer that you mostly launch from the command line, it's also compatible with both X11 and Wayland. This means it's going to be a perfect utility as we hope to move from the increasingly decrepit X11 to the new Wayland display system.

Despite stability still being some way off, tentative Wayland support can already be found in *GTK* and *Qt*, and subsequently, Gnome and KDE, and its slimmed down remit and high-performance code promises to revolutionise desktop performance.

Not only can *imv* display images using Wayland, it can display animated gifs and many different image formats including RAW and *Photoshop* PSD files. It can overlay information about an image and change the way it scales an image to fit. Thanks to the command line, you can get lots of added functionality for free. You can give it a wildcard, such as ***.jpg**, and jump between a collection of images using the cursor keys to create an *ad hoc* slideshow, for instance, or use the output of the **find** command to display an image without knowing its location.

Cleverly, you can also use the P key to send the current image to the standard output, which lets you construct commands that will automatically email the selected image or change your desktop background according to whichever image you choose. Most importantly, it's the quickest image viewer we've used for years, even with large images, and isn't tied to any particular desktop or toolkit. It would be ideal on a low-powered device such as a Raspberry Pi, especially as it's likely that Wayland will run must faster on the Pi and will need an image viewer.



```
[graham@mbp-arch photos]$ imv *.jpg
```

Imv is the first image viewer we've found that's developed to work on the new windowing system, Wayland

PROJECT WEBSITE
https://github.com/eXeC64/imv

FOSSPICKS Brain Relaxers

First person shooter

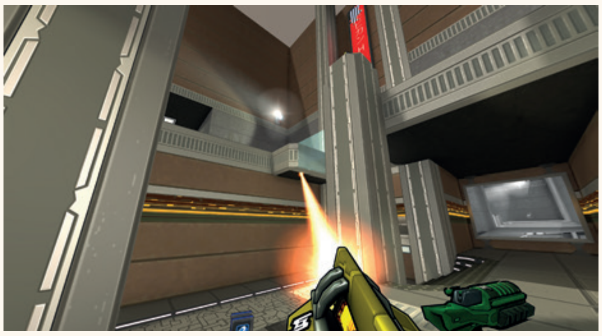
Warsow 2

We know there are still people devoted to *Counter-Strike* after 15 years; people who flinch when a fizzy drink can is opened thinking it's a grenade. People like these still meet virtually every week and chat over *Teamspeak* while they plan their next conquest. This doesn't quite describe us; instead, we enjoy first person shooters for what they offer the casual gamer – an intense rush of adrenaline.

No other type of game can bring a group of strangers together to defend a pretend flag, or run from one side of the map to the other while trying to target an almost identically equipped opposition. They can be great fun, which is why they're still a major part of the games industry.

Linux has had its fair share of classics, from *Unreal Tournament* to the more recent *Team Fortress 2*. But there's more to this comparison than two different first person shooter types – *Unreal Tournament* was a stalwart of the old Linux games scene before *Unreal Tournament 3* failed to make its way to Linux in 2007, despite years of vague promises that followed. *Team Fortress 2* was one of Valve's flagship titles when it started its great SteamOS campaign, turning Linux into a games console and changing many attitudes towards Linux games development almost overnight.

Between those two games, the vacuum of commercial releases was taken up by the open source community. *Alien Arena* and *Red Eclipse* are two open source shooters that we love to play, but


Unless they cripple you with motion sickness, first person shooters like *Warsow* are brilliant fun.

we've always had a soft spot for *Warsow*, which has just celebrated a major milestone with the release of 2.0.

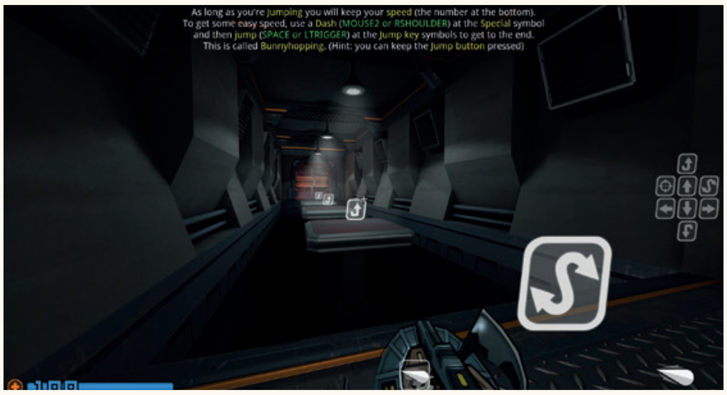
Warsow is similar in playing style to the much-loved Unreal Tournament

Atrocity Exhibition

Despite our always mistaking the title with the capital of Poland (or the prototype Joy Division), *Warsow* is a brilliant FPS, similar in playing style and frenzy to the much loved *Unreal Tournament*. Its combat is well enough established that even those old *Counter-Strike* players will recognise its tournament potential. There's a great tutorial to ease you into the game, and you can then join what of the several games that always seem to be running, or create your own local instance for friends and colleagues to join.

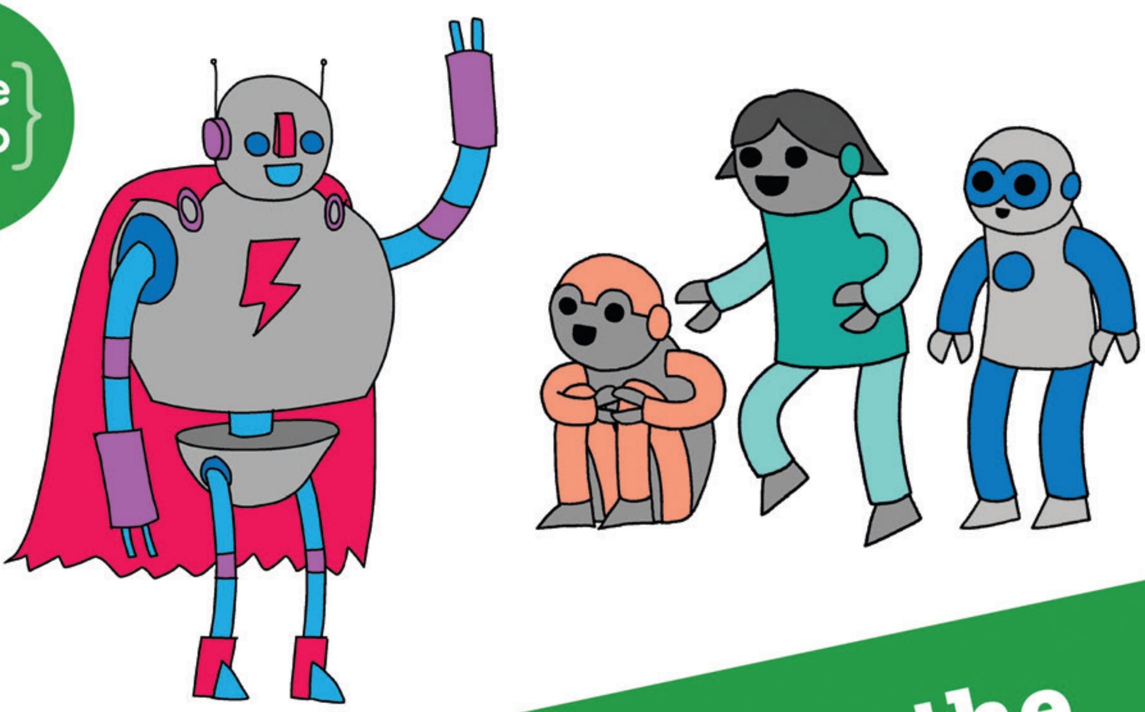
The cell shading effect used for the graphics looks fantastic, even on modest modern hardware running at a high frame rate.

There's also more to the game than shooting, with some excellent platform elements such as double-jumps and wall jumps, and thanks to the JavaScript-based scripting engine, there are lots of different game types. The icing on the cake is that unlike many other open source games, nearly all the media assets are released under the terms of the Creative Commons, making this an exceptionally open FPS.


As long as you're jumping you will keep your speed (the number at the bottom).
To get some easy speed, use a Dash (MOUSE2 or RSHOULDER) at the Special symbol
and then jump (SPACE or LTRIGGER) at the Jump key symbols to get to the end.
This is called Bunnyhopping. (Hint: you can keep the Jump button pressed)

Top: Even if you've never played a game like this before, *Warsow*'s excellent beginner's tutorial will guide you through the basics of control and combat.
Bottom: The combat in *Warsow* is exceptionally fast, and relies on quick reactions and mastery of the weapons on offer.

PROJECT WEBSITE
www.warsow.gg



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**

This advertising space was kindly donated to Code Club by Linux Voice

TUTORIALS

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

Ben Everard
believes in a future where all technology is built to make us happy.

Working on a magazine about Linux, you have to assess a huge range of technology. I've come to the conclusion that the only thing that matters when deciding if a device is good or bad is the answer to the question 'Does it improve my life in any way?' Here are the three bits of tech that improve my life the most.

- An Intel CPU with integrated 3D graphics. I install a lot of distros, and (even without taking free software concerns into account), life's too short to have to worry about proprietary drivers.
- An Arduino Uno. This microcontroller opened up a whole world of tinkering with programmable circuits that I thought was beyond my reach. Plugging your own circuits into a USB port and running custom code on them is my view of geek heaven.
- Over-ear head phones. They they make listening to Linuxy podcasts while shopping in Tesco far more enjoyable.

I would encourage you to think about which bits of tech you have that improve your life the most to help you make better choices in the future.
ben@linuxvoice.com

In this issue . . .

Get access to every Linux Voice tutorial ever published in our digital library of back-issues available exclusively to subscribers – turn to page p56 to join.

FOG: CLONE COMPUTERS OVER THE NETWORK

Walk the fine line between being lazy and slick with the agility of a tightrope walker.

MAYANK SHARMA

WHY DO THIS?
- Deploy any number of machines without any hassle
- Easily image critical installations to minimise downtime
- Works with heterogeneous networks comprising Linux, Windows and OS X

Managing a lab full of computers can be quite tiring. The constant barrage of repetitive tasks can sap the energy out of any sysadmin, irrespective of the size of their realm. Thank heavens then for the *Fog* project, takes the pain out of regular admin tasks such as installing software, and can even manage printers on the network.

The *Fog* server is scalable and can manage large networks spread over multiple locations in the same building or on the other side of the planet. One of the most useful features of the *Fog* server, especially for admins of larger networks, is the multicast ability. Using this feature you can deploy multiple machines in one go. To supplement it on such large networks, you can have multiple *Fog* installations configured as storage servers that help take the load of the main *Fog* server when imaging computers. However, *Fog*'s most essential task is to image an installation and to deploy it to other computers on the network, which is what we'll cover in this tutorial.

STEP BY STEP: IMAGE AND CLONE A COMPUTER

1 Set up the image server

Before installing *Fog*, make sure the server has a static IP address, which can be easily ensured from your router's admin page. Also make sure that all the machines in your network are configured to boot from the network card. Finally, remember to disable any existing DHCP servers on the network, as we'll set up the *Fog* server as a DHCP server and dole out addresses to all the computers on the network.

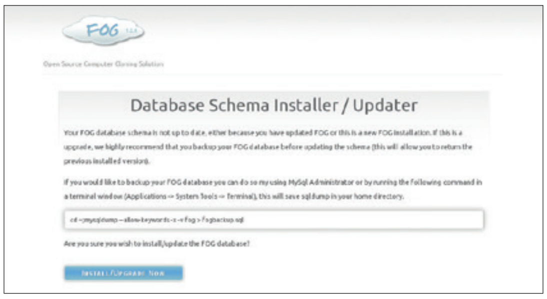
Once you have your network set up, head to the machine that you've earmarked as the *Fog* server and download the latest stable *Fog* release from SourceForge (**http://sourceforge.net/projects/freeghost/files/FOG**). Then fire up a terminal and extract the downloaded tarball with

```
tar zxvf Fog_1.2.0.tar.gz -C /opt
```

Change into the **bin/** directory under the extracted tarball, and fire up the installation script with

```
sudo ./installfog.sh
```

The installation script will prompt you for several bits of information. They are self-explanatory, and in most cases it's best to go with the default options.



Database Schema Installer / Updater

Your FOG database schema is not up to date, either because you have updated FOG or this is a new FOG installation. If this is a upgrade, we highly recommend that you backup your FOG database before updating the schema (this will allow you to return the previous installed version).

If you would like to backup your FOG database you can do so my using MySql Administrator or by running the following command in a terminal window (Applications -> System Tools -> Terminal), this will save sqldump in your home directory.

```
cd ~;mysqldump --allow-keywords -x -v fog > fogbackup.sql
```

Are you sure you wish to install/update the FOG database?

2 Create base image

After it's fetched and installed any required components, the installation script will display a URL for *Fog*'s dashboard. Open the link in your web browser and log in with the default credentials (**fog:password**). Before going further head to User Management > Create New User to define a new administrator.

To begin the process of imaging a computer, head to Image Management > Create New Image. Use the fields in the form to describe the image. For example, let's assume we're creating an image of an OpenSUSE 42.1 installation that we'll then use on all our workstations. We can name the image 'Workstations/Desktops' and use the Operating System pull-down menu to specify the operating system of this image, that is Linux. Finally, select the correct disk layout scheme from the Image Type pull-down menu. Our OpenSUSE installation is on a single disk with multiple partition so we'll select the second option.

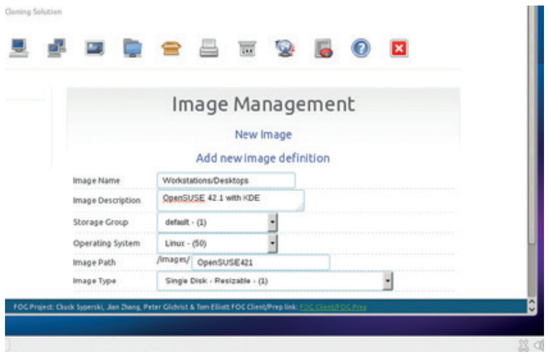


Image Management

Add new image definition

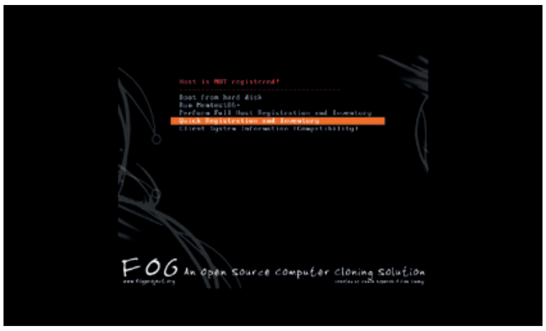
Image Name	Workstations/Desktops
Image Description	OpenSUSE 42.1 with KDE
Storage Group	default - (1)
Operating System	Linux - (50)
Image Path	/images/ OpenSUSE421
Image Type	Single Disk - Resizable - (1)

FOG Project: Chuck Syperski, Jian Zhang, Peter Gilchrist & Tom Elliott FOG Client/Prep link: FOG Client/FOG Prep

3 Register host and associate image

Now head to the computer with OpenSUSE 42.1 that you wish to use as the base image and boot it up. Since the computer is set to boot from the network card, it'll display the PXE boot environment from the *Fog* server. Scroll down the *Fog* menu and select the 'Quick Registration and Inventory' option. The *Fog* server will now scan the computer and add it to its repository of known hosts.

When it's done, power down the OpenSUSE computer and head back to the *Fog* server. Fire up the dashboard and head to Host Management > List All Hosts. You should now see the OpenSUSE machine listed here, which by default is identified by its MAC address. Click on the edit icon to change it to something more identifiable, like 'OpenSUSE 42.1'. Most importantly, use the Host Image pull-down menu and select the Workstations/Desktops option for the image you created earlier.

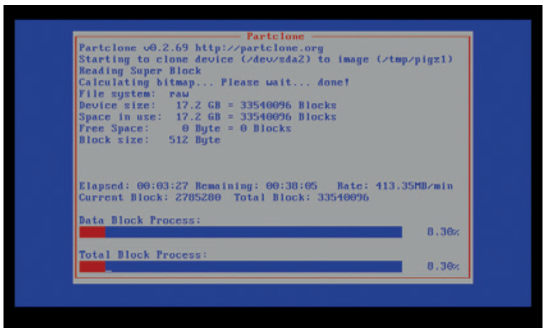


```
Host is NOT registered!
------------------------------------------
Boot from hard disk
Run Memtest86+
Perform Full Host Registration and Inventory
Quick Registration and Inventory
Client System Information (Compatibility)
```

4 Image the host

We're now all set to image the OpenSUSE installation. Head to Task Management > List All Hosts, which will list the recently added OpenSUSE 42.1 machine. Under the Task section, click on the green upload arrow corresponding to this image. *Fog* will give you multiple options to schedule the upload task. You can tinker with these in the future, but for now it's best to go with the default option for instant deployment.

Now head to the OpenSUSE machine and boot it up. It'll again detect *Fog*'s PXE and automatically image the machine and upload it to the *Fog* server. The process will take some time depending on the size of the disk it has to image, the processing capabilities of the computers involved and the speed of the local network. The OpenSUSE computer will restart once it's done uploading the image. That's all there's to it. Repeat steps 2–4 to similarly image any other computer on the network.

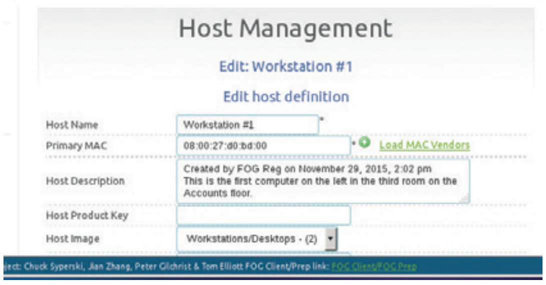


```
Partclone
Partclone v0.2.69 http://partclone.org
Starting to clone device (/dev/sda2) to image (/tmp/pigz1)
Reading Super Block
Calculating bitmap... Please wait... done!
File system:  raw
Device size:   17.2 GB = 33540096 Blocks
Space in use:  17.2 GB = 33540096 Blocks
Free Space:     0 Byte = 0 Blocks
Block size:   512 Byte


Elapsed: 00:03:27 Remaining: 00:30:05   Rate: 413.35MB/min
Current Block: 2785280  Total Block: 33540096

Data Block Process:
                                                   8.30%
Total Block Process:
                                                   8.30%
```

5 Register target machines

Before you can deploy an image to another computer, you need to first register it with the *Fog* server. The registration process is the same as before. Boot the new computer from the network which should detect *Fog*'s PXE environment. And when it does, select the 'Quick Registration and Inventory' option.

Once the computer has been added to *Fog*'s repository of known computers, log in to the *Fog* dashboard and head to Host Management > List All Hosts. Just like before, click on the edit icon corresponding to the newly added machine and rename it so that it's more identifiable, and associate the Workstations/Desktops image with this computer using the Host Image pull-down menu. Repeat the process to register all the computers with the *Fog* server, then edit them in the *Fog* dashboard to give them a name and associate them with the appropriate image.



Host Management

Edit: Workstation #1

Edit host definition

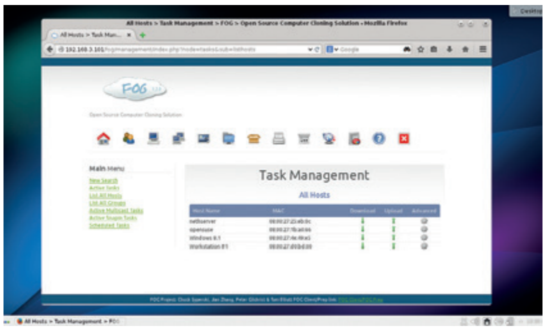
Host Name	Workstation #1
Primary MAC	08:00:27:d0:bd:00
Host Description	Created by FOG Reg on November 29, 2015, 2:02 pm This is the first computer on the left in the third room on the Accounts floor.
Host Product Key	
Host Image	Workstations/Desktops - (2)

ject: Chuck Syperski, Jian Zhang, Peter Gilchrist & Tom Elliott FOG Client/Prep link: FOG Client/FOG Prep

6 Deploy the image

Now to replicate the OpenSUSE image onto the other computers, head to Task Management > List All Hosts. Browse the list of hosts to find the entry for the computer to which you wish to deploy and hit the corresponding down arrow Download image button. Now head to this workstation and power it on. The computer's PXE environment will automatically detect the task from the *Fog* server and begin copying the image from the server on to the local machine. When it's done, you'll end up with a mirror copy of the OpenSUSE 42.1 installation on this workstation.

Besides deployment tasks, you can create various other types of tasks to check up on the computer and its installation. Click on the gears icon to bring up a list of several deployment options, such as Test Disk or Password Reset. Select one and then power on the machine to automatically launch the task.



Task Management

All Hosts

SQUID: CONTROL WEB BROWSING BY PROXY

Shield your browser from the ravages of the web and use a proxy as protection.

BEN EVERARD

WHY DO THIS?
- Save bandwidth
- Block web adverts
- Be more productive

A proxy is something (or someone) that performs an action on your behalf. In the case of a web proxy such as *Squid*, it fetches web pages for you. If you configure your browser to connect to a proxy rather than the internet, it doesn't download the pages directly. Instead, it sends a message to the proxy indicating which pages are needed, and the proxy gets them and sends them to the web browser.

This may all sound very pointless, but by having a proxy between your browser and the web, you can fine-tune the way you connect. Since many machines can share a proxy (it doesn't have to be running on the same machine that uses it), you can quickly configure the way your web connections work across a range of devices.

Squid is the most popular proxy for Linux, and is available in most distros' package managers in a package called **squid** or **squid3**. Before jumping in and installing *Squid*, it's worth thinking a little about what machine you want to install it on. If you only want to use a proxy on a single machine, you may as well just install it on that machine. However, as we said before, you get the best out of *Squid* when it's shared between many computers. In order for this to work, you need to have a computer that's usually turned on and attached to your local network. If you've already got a home server, that's the ideal machine to use as a *Squid* proxy. It's really easy to copy the configuration over from one machine to another, so if you're not sure, you can just install it on your main machine to try it out, and then change over to a different machine later if you find it useful.

As many machines can share a proxy, you can configure the way your web connections work on several devices

Once you've installed *Squid* from your package manager, you'll need to make sure the service is started. The method for this varies a little, but the following will work on most modern distros:

```
sudo service squid3 restart
```

You can check that it's running correctly by trying to connect your web browser. In *Firefox* this is in Edit > Preferences > Advanced > Network > Settings. Select Manual Proxy Configuration and enter the HTTP Proxy as **localhost** (or wherever you installed *Squid*) and the Port as 3128. Press OK to accept the settings. If you use *Chrome* or *Chromium*, you'll have to set the proxy details at the OS level. The method for this differs between distros and desktops, but should be possible in your network settings app.

If you can still view web pages, then everything's worked. You can also double-check that everything's going through the proxy by looking in the access log. You can do this with:

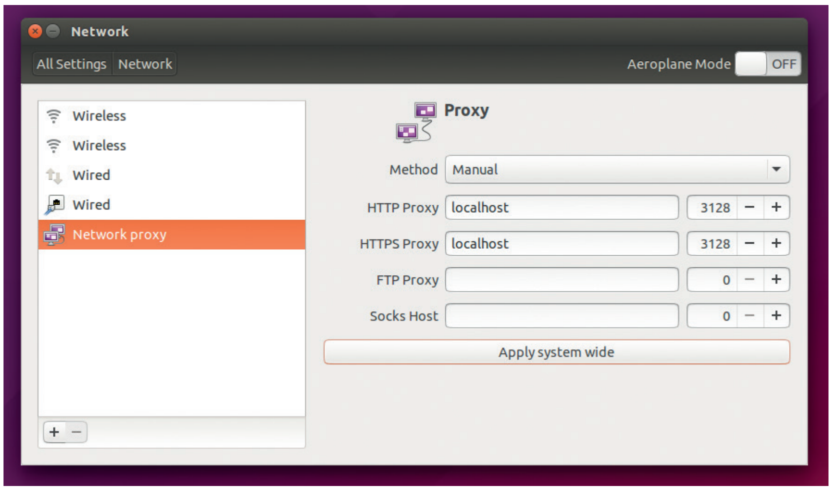
```
sudo tail /var/log/squid3/access.log
```

Distros other than Debian or Ubuntu may store the log in a different place.

That's all there is to getting a web proxy running on Linux. However, to really get the most our of your proxy you need to configure it to your needs.

Getting personal

Traditionally, the most common use for a proxy has been to reduce the bandwidth used, by sharing a temporary store. If one person connected to the proxy requests, for example, **www.linuxvoice.com**, then the proxy fetches this from the internet and passes it on. If a second person requests the same page, the proxy already has the data for this site, so it doesn't need to request it from the website again: it just passes it straight from the proxy cache to the second person's



Method: Manual
HTTP Proxy: localhost 3128
HTTPS Proxy: localhost 3128
FTP Proxy: 0
Socks Host: 0

Configuring your proxy at OS level will ensure that it gets picked up by all your software that connects to the internet.

web browser. The advantage of this varies a lot depending on who's using the web connection. If you're managing a corporate or school network, there's a good chance that you could save quite a bit of bandwidth. If you're managing a home network for just one or two people then the benefits are likely to be less, but it can still be worth doing especially if it's not as fast a connection as you'd like.

By default, *Squid* will only proxy files in 256MB of memory. This isn't very much, so it probably won't have a noticeable effect on your web browsing. There are two options to increase it: you can increase the amount of memory available or you can configure *Squid* to use the hard drive for the cache. Both of these are configured in the **squid.conf** file, which is usually found in **/etc/squid3**.

To make more memory available (for example, 1GB), open this config file with your favourite text editor, such as with:

```
sudo nano /etc/squid3/squid.conf
```

and find the line:

```
# cache_mem 256 MB
```

The hash at the start of the line means that it's commented out; however this is also the default amount, so deleting the hash won't immediately change anything. Instead, change the line to give a different amount of memory to the proxy such as:

```
cache_mem 1024 MB
```

To change the disk caching, find the line that starts with:

```
# cache_dir
```

The options on this configuration line allow for very fine-tuned control over the cache. The first argument is the storage method to use. There are several options that each have different payoffs in terms of space efficiency and time efficiency and other aspects, but unless you're running a really high-traffic proxy, the default of **usf** (Unix File System) should work fine.

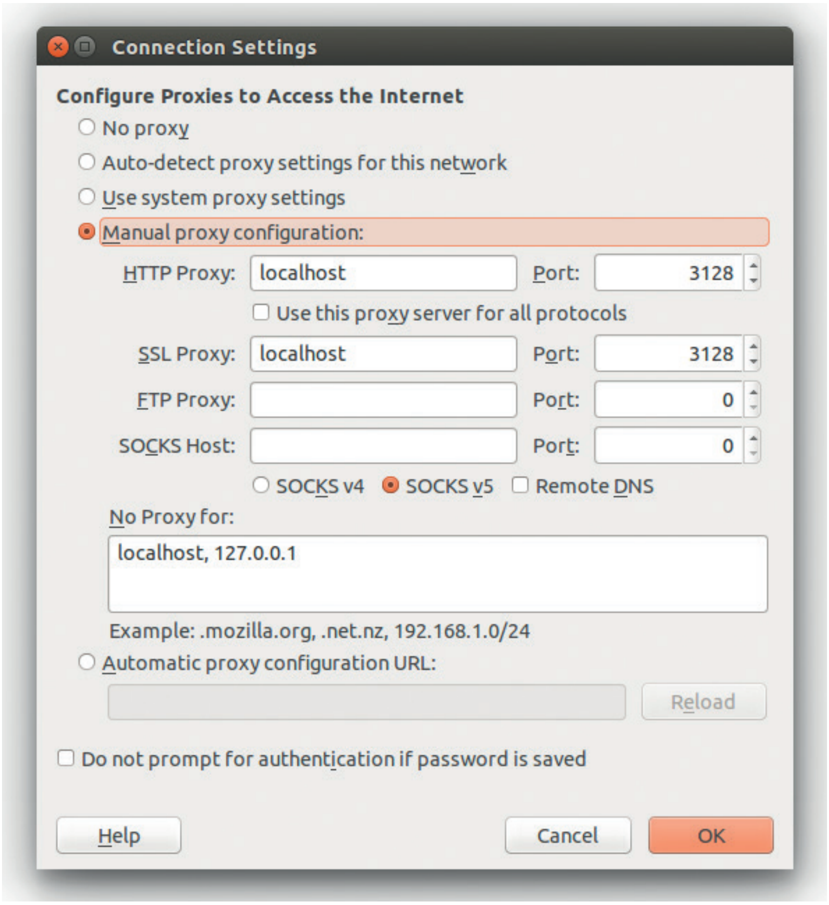
The second argument is the location of the cache. It will be a directory structure rather than just a single file, and the default location is in **/var/spool**, though anywhere to which the *Squid* user has write access is fine. Following this are the arguments for UFS. These are three numbers, the first of which is of most interest to us, as it's the total amount of space that *Squid* can use on the disk. The second number is the number of subdirectories that can be in the cache root, and the third number is the number of subdirectories allowed inside the first set. The defaults for these two are fine for most uses. To create a 1GB disk cache, change the line to the following:

```
cache_dir ufs /var/spool/squid3 1024 16 256
```

Once you've saved your changes, you just need to tell *Squid* to reload the configuration file with:

```
sudo squid3 -k reconfigure
```

Advertising provides a good way for a website to make a little money to cover the cost of hosting the site. Popular websites can also earn enough to pay people to create content, so in this regard, they're something that we all benefit from. However, through 2014 and 2015 there was a massive increase in invasive advertising, where adverts blocked large portions of the site and aggressively tracked people on the site in an attempt to squeeze pennies out of them. The result is that now a sizeable proportion of bandwidth and CPU power is spent on rendering adverts that the viewer didn't want in the first place.


Connection Settings

Configure Proxies to Access the Internet

No proxy / Auto-detect proxy settings for this network / Use system proxy settings / [✓ Manual proxy configuration:]

HTTP Proxy: localhost Port: 3128

☐ Use this proxy server for all protocols

SSL Proxy: localhost Port: 3128

FTP Proxy: Port: 0

SOCKS Host: Port: 0

SOCKS v4 [✓ SOCKS v5] ☐ Remote DNS

No Proxy for: localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

☐ Do not prompt for authentication if password is saved

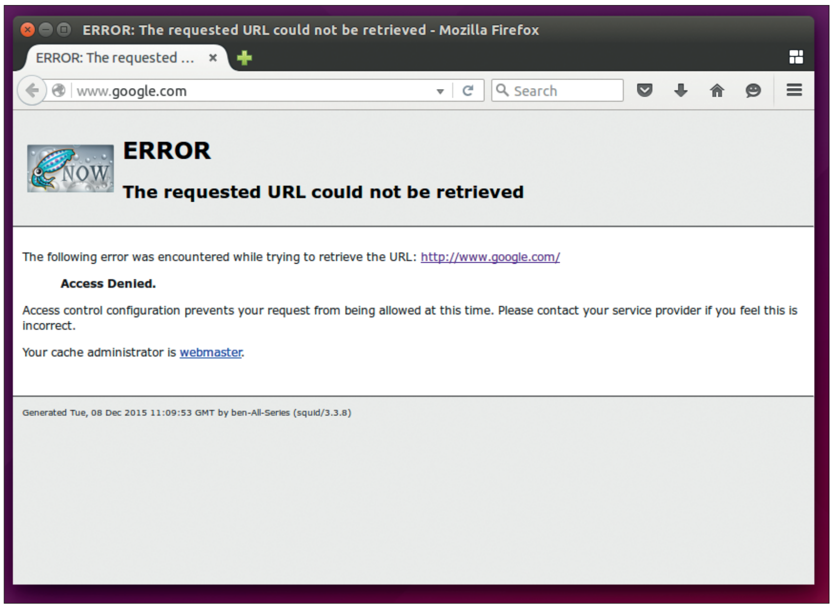
The *Firefox* configuration needs both HTTP and SSL proxies set to capture all traffic.

Admonish ads

There are already a number of options for blocking adverts, including web browser plugins; however, many of these plugins are themselves nefariously tracking users and feeding this data back to the advertisers. If you run a network either at home or work, you probably also have a number of devices – and wouldn't it be much easier if all of them could have their adverts blocked at a central point? If you do it with your proxy, every machine that connects through the proxy automatically has adverts removed.

Squid uses Access Control Lists (ACLs) to decide which traffic to let through and which to block, so to block adverts, we just need an ACL that will identify which bits of traffic are adverts, and block these. In web access lingo, this is termed a blacklist. The opposite of a blacklist is a whitelist, which contains details of traffic that we do want to let through.

Blacklists are available online both for free and commercially, and which one is right for you will


ERROR: The requested URL could not be retrieved - Mozilla Firefox

ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: http://www.google.com/

Access Denied.

Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is webmaster.

Generated Tue, 08 Dec 2015 11:09:53 GMT by ben-All-Series (squid/3.3.8)

By default, users visiting blocked sites will be served an error page. This is configurable in **squid.conf**.

depend on what level of blocking you need. If you're blocking adverts (like we are here), a free blacklist is probably sufficient, since it's not a problem if the occasional advert slips through. If you find yourself relying on *Squid* to enforce network policies, then it's worth investigating further options.

We'll use the blacklist from **http://pgl.yoyo.org/as/**. It's quite comprehensive and uses regular expressions to keep up to date with changing subdomains. You'll need to select the type as 'Squid – As Squid Dstdom_regex File', tick the View List option as Plain

> You may want to block Facebook during weekdays, and only allow access in the evening and at weekends

Text and then press Go. This will open the file in your browser, so save it to your hard drive with the name **advertdomains**.

You can configure *Squid* to look anywhere you like for this file, but in the interests of keeping everything contained, it's best to place it in the *Squid* config directory, so copy it and change the owner of the file to the *Squid* user with:

```
sudo cp advertdomains /etc/squid3/advertdomains
sudo chown proxy /etc/squid3/advertdomains
```

This should work on Ubuntu- and Debian-based systems. If you're on a different distro, you may need to change the location of the config directory and the username of the *Squid* user. Check your distro's documentation if you have issues with this.

Now you need to open **squid.conf** in your favourite text editor and add the lines to load and act on the blacklist:

```
acl ad-domains dstdom_regex "/etc/squid3/
advertdomains"
http_access deny ad-domains
```

The first line creates an access control list called **ad-domains** of the type **dstdom_regex** and loads the data from the file **/etc/squid3/advertsdomains**. The quote marks around the file name are important, because without them, *Squid* will attempt to interpret the file location rather than the file contents as the blacklist data. The second line creates a rule for **http_access**. If you wanted to create a whitelist rather than a blacklist, you could use **allow** rather than **deny**.

Once you've saved those changes, you can reload the *Squid* configuration using:

```
sudo squid3 -k reconfigre
```

Now adverts should be blocked on all browsers that are routing through the proxy.

Access denied

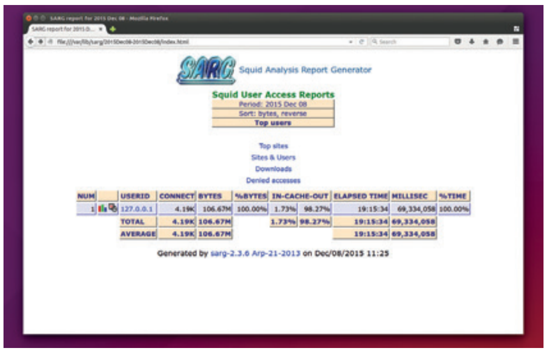
Our above method of blocking adverts was easy to set up because we had a ready-made blacklist in the right format. However, if you want to block other categories of content, you won't always find blacklists in formats that *Squid* can understand. Fortunately, Linux gives us plenty of text processing tools that we can use to transform everything we need into a format that *Squid* can work with.

There's a great set of blacklists put together by Shalla Secure Services are available at **www.shallalist.de**. This website has a download link that will grab a tarball you can extract to get blacklists for a range of different categories of material. In each subfolder of the main **BL** folder, you'll find a **domains** text file containing one domain per line of content that should be avoided in this category. There are two problems with this. First, *Squid* won't automatically exclude subdomains unless the domain is preceded with a dot; and second, *Squid* will throw an error if the list includes both a subdomain and a higher domain preceded by a dot. If we want to use the Shalla list, we need to remedy these two problems. Open a terminal and navigate to the subfolder you want to ban.

To add a leading dot to all the lines, use the following command, which matches the start of a line character (**^**) and inserts a dot:

```
cat domains | sed 's/^/./' >domainsquid
```

Solving the second problem is a little trickier. We need to remove any lines that are subdomains of domains that are included in the file. The first task


Squid Analysis Report Generator

Squid User Access Reports

Period: 2015 Dec 08

Sort: bytes, reverse

Top users

Top sites

Sites & Users

Downloads

Denied accesses

NUM		USERID	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME
1		127.0.0.1	4.19K	106.67M	100.00%	1.73%	98.27%	19:15:34	69,334,058	100.00%
		TOTAL	4.19K	106.67M		1.73%	98.27%	19:15:34	69,334,058	
		AVERAGE	4.19K	106.67M				19:15:34	69,334,058	

Generated by sarg-2.3.6 Arp-21-2013 on Dec/08/2015 11:25

Sarg (Squid Analysis Report Generator) will parse your *Squid* logfile to create a report to help you understand how your internet connection is used.

here is to make sure that the file is sorted so that all the subdomains are listed underneath the original domains. A normal sort won't help us here, because the subdomains are on the left of the domain, and **sort** organises a list using the leftmost character. The easiest way around this is just to reverse each line before the sort, and then reverse again at the end. We can do this using the **rev** command. Once this is done, we can use **awk** to match the domain against the same number of characters on the line below. If the two match then the second line is a subdomain of the first line and it shouldn't be output. All together, this is done with the following command:

```
rev domainsquid | sort | awk 'NR!=1&&substr($0,0,length(p))==p{next}{p=$0".";print}' | rev > newdomainsquid
```

This list is a **dstdomain** type in *Squid*, so you need to add the following lines to your **squid.conf**:

```
acl blocksites dstdomain "/etc/squid3/newdomainsquid"
http_access deny blocksites
```

When you reload the *Squid* configuration, this will block all the domains listed in **newdomainsquid**. You can have as many of these blacklists as you like provided you give each of them a different name (**blocksites** is the name here).

Fine tuning

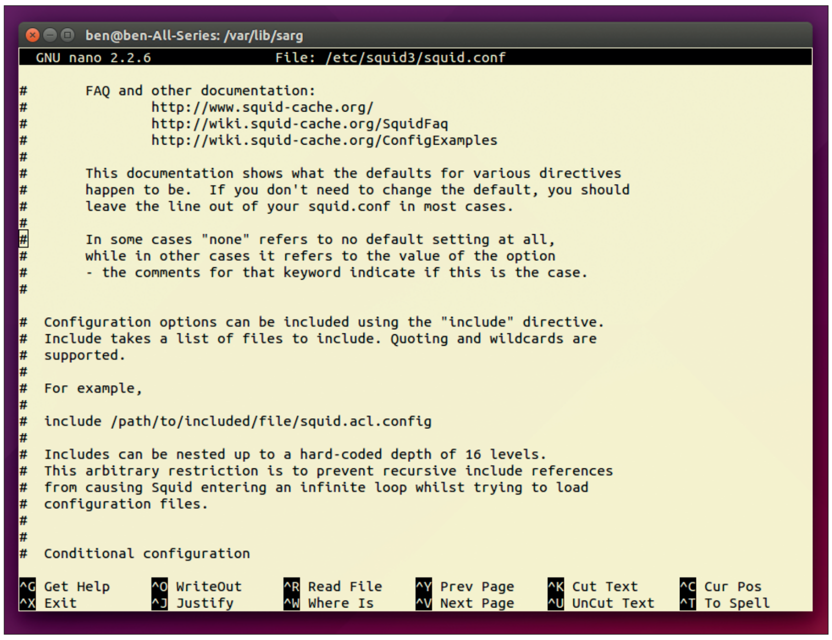
Access Control Lists enable more nuanced control than simply allowing or blocking. One of the most useful alternatives is the use of time-based controls, which enable you to alter which sites are blocked based on the time of day or day of the week. For example, you may want to block the Facebook website during weekdays, and only allow access in the evening and at weekends. You could do this by adding the following to your **squid.conf** file.

```
acl facebooktime time MTWHF 18:00-23:59
acl facebookdomain  dstdomain .facebook.com

http_access allow facebookdomain facebooktime
http_access deny facebookdomain
```

The first line defines an ACL with the **time** type, and this takes two arguments. The first argument is the list of days to match on (this is the first letter of the day, except for Thursday, which is a H, and Saturday, which is an A). The second argument is the time period to allow.

The first **http_access** line combines two ACLs and only lets a web request through if it matches both of the ACLs. We also need the final line to deny other requests to this domain. In this example, we've just used a single domain, but exactly the same approach could be used with one of the domain lists from Shalla to (for example) block all web chat at certain times.

Squid's **http_access** doesn't allow a way of combining ACLs so that access is granted if either one or the other ACLs is matched; instead, this can be done by creating a **http_access allow** line for each allowable combination of ACLs, followed by a **http_access deny** line. Facebook is served over HTTPS, so in order for this block to work, you have to make sure that your browser is configured to send encrypted pages to the proxy as well as unencrypted ones. In *Firefox*, this is set in the proxy configuration page on the SSL line. LV



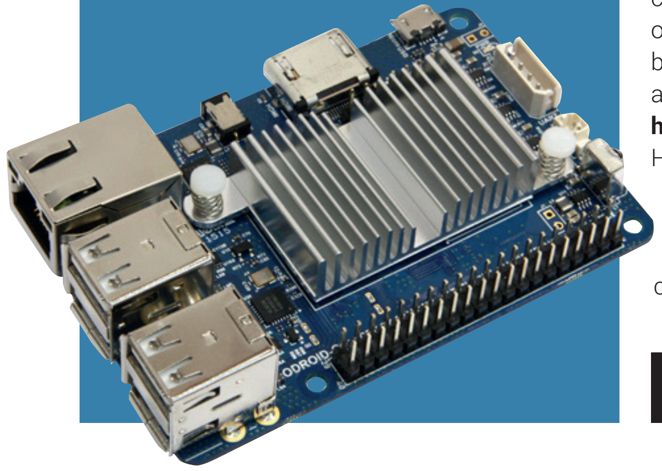
```
ben@ben-All-Series: /var/lib/sarg
  GNU nano 2.2.6              File: /etc/squid3/squid.conf

#       FAQ and other documentation:
#               http://www.squid-cache.org/
#               http://wiki.squid-cache.org/SquidFaq
#               http://wiki.squid-cache.org/ConfigExamples
#
#       This documentation shows what the defaults for various directives
#       happen to be.  If you don't need to change the default, you should
#       leave the line out of your squid.conf in most cases.
#
#       In some cases "none" refers to no default setting at all,
#       while in other cases it refers to the value of the option
#       - the comments for that keyword indicate if this is the case.
#

#  Configuration options can be included using the "include" directive.
#  Include takes a list of files to include. Quoting and wildcards are
#  supported.
#
#  For example,
#
#  include /path/to/included/file/squid.acl.config
#
#  Includes can be nested up to a hard-coded depth of 16 levels.
#  This arbitrary restriction is to prevent recursive include references
#  from causing Squid entering an infinite loop whilst trying to load
#  configuration files.
#
#
#  Conditional configuration

^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

The **squid.conf** file includes detailed comments about all the options, so you should be able to get things working even if you've broken your internet connection.

Running a dedicated proxy server

The setup we've done here has been based on running your proxy locally, but if you want your proxy to be available to all devices on the network, you'll need a machine that's always on (or, at least, is turned on as often as you want the network to be on). Obviously, when choosing a machine to be turned on constantly, power consumption is an issue, so any of the small ARM boards (such as a Raspberry Pi or Odroid) make an excellent choice. Another option is to run *Squid* directly on your network router. Most ISP-supplied router firmware doesn't come with this option, but if your router is capable of running OpenWRT (**https://openwrt.org**) you can run *Squid* on the same box that that handles your main network connection.

Other than running the machine, the configuration is exactly the same as we've described here, so you just need to copy across your **squid.conf** file and any associated black and whitelists, and *Squid* will be ready to run.



Ben Everard intends to monetise *Squid* by selling Daily Mail blockers for people who need to keep their minds clean.

BUILD A GAME WITH GPIO ZERO AND SCRATCH

Remember Operation? Of course you do – now make your own!

LES POUNDER

WHY DO THIS?

- Control hardware using the GPIO
- Learn logic
- Transfer knowledge across languages

TOOLS REQUIRED

- A Raspberry Pi running the latest Raspbian release
- 3 x LED
- 3 x 220Ω resistors
- Male–female jumper cables
- Male–male jumper cables
- A buzzer
- Breadboard
- Masking Tape
- Aluminium Foil
- Wire
- Glue
- A plastic case (we used an A4 document holder)

For this issue we are going to create one hardware project, in this case a homemade "Operation" game where we have to save the robot. We have three lives and if we touch the metal of the robot we lose a life, which is indicated by one of three LEDs turning off and the buzzer sounding. This project uses a simple method of input: the surgery tool is turned on with current, and when we touch the foil around the robot, we connect to Ground, causing the tool to turn off, triggering the code to execute.

To code this project we'll use two methods, each one aimed at a different level of user. For example, beginners can learn to hack with Scratch, whereas more competent coders can use the new GPIO Zero Python library, which removes a lot of the hassle of using the old RPi.GPIO Python library.

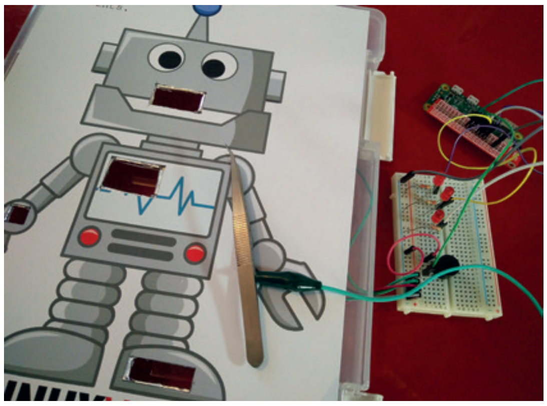
The circuit diagram for this project, along with the full code listings and other images, can be found via our GitHub repository at **https://github.com/lesp/LV24-Dr-Robot**, or you can download a Zip file containing all of the project files from **https://github.com/lesp/LV24-Dr-Robot/archive/master.zip**.

PROJECT 1 – SCRATCH

You'll find Scratch in the Programming menu of the latest version of Raspbian (Jessie). To start coding we shall use blocks from the left of the screen and drag them into the centre coding area. We start with a hat block: in the Control palette, Look for "When Green Flag Clicked" and drag it into the coding area. Still in the Control palette look for "broadcast"; we're going to create seven broadcast blocks, each used to configure the GPIO.

Our first turns on the GPIO server, a script behind the scenes that enables Scratch to talk to the GPIO. We are next going to create four configurations for GPIO pins numbered 17,27,22 and 10, turning them into outputs. We configure GPIO 9 to be an input for our surgery tool, before finally we create a broadcast to turn on three LEDs – more on that later. So you should have

> Beginners can learn with Scratch, while more competent hackers can use the new GPIO Zero Python library


Our finished project is a mix of arts materials, electronics and aluminium foil circuits and uses many different skills across the curriculum.

```
When Green Flag Clicked
broadcast gpioserveron
broadcast config17out
broadcast config27out
broadcast config22out
broadcast config10out
broadcast config9in
broadcast 3LED
```

Now let's create a new section of code. Grab another When Green Flag Clicked block from the Control palette, then a Forever loop and attach it to the Green Flag block. In the Control palette you will see Repeat 10 – drag it inside the Forever loop and change 10 to 2, as we're going to sound the buzzer attached to GPIO 10 twice for half second intervals, add the following blocks.

```
broadcast gpio10on
wait 0.5 secs
broadcast gpio10off
```

Now we come out of the Repeat 2 loop, but still inside the Forever loop. Drag a Repeat Until loop from the Control palette so that it is under the Repeat 2 loop, and inside the Forever loop. We now need to create a variable, so go to the Variables palette and create a variable for all sprites called Lives. Once it's created, drag the Lives block to the coding area and put it somewhere safe. Now from the Operators palette drag the __ = __ block and place it inside the blank space at the top of the Repeat Until block. In the right-hand blank space of __ = __ type the number zero, and in the left drag the Lives variable and drop it inside. Now any code inside this section will run until the user runs out of lives.

We next need to drag an If block, placing it inside the Repeat Until block. We also need another __ = __ in the blank space of the If block. Add a zero (0) to the right-hand side of the __ = __ block. For the left-hand side we need to read the state of our surgery tool, which will be 1 (on) or when touching the foil, 0 (off). This block will be in Sensing, but first we need to activate our code to register the GPIO pins, so click on the Green Flag in the top-right of your screen. Now return to Sensing and look for the Slider Sensor Value block. Click on the drop-down and you'll see GPIO9. Change the block to GPIO9 and drag it into the left blank of __ = __.

Inside this If condition we'll create another Repeat loop, this time for three iterations. Each time it will turn on GPIO 10, wait for 0.2 seconds, and then turn it off before waiting and repeating the sequence.

Breaking out of the Repeat 3 loop but still inside the If condition we now need to change the value of the Lives variable by -1. In the Variables palette you will see "change lives by 1" – drag it under the repeat 3 and change the value to -1. Now drag a "Say for 2 secs" block from the Looks palette and use it to tell the player they have lost a life.

Under the Say block we now place an If condition that will compare the value of lives against an integer, we will need the __ = __ block from Operators and the Lives variable, which goes in the left __ = __ (in the right-hand side, type "2"). Inside the If condition we will create a new broadcast block called 2LED. Repeat these steps for 1LED and 0LED. Place each If condition under one another.

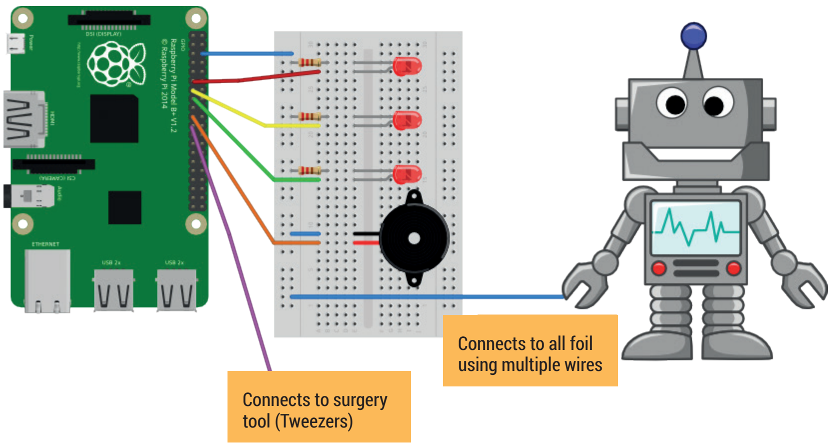
We now break out of the "Repeat Until Lives" loop but stay in the Forever loop. We drag another "Say for 2 secs" and use that to say Game Over; we next change the Lives variable so that we have 3 lives. Lastly we create a broadcast "3LED".

Remember all those 0LED, 1LED broadcasts that we created earlier? Now we're going to create sequences of code that respond to those broadcasts. In the Control palette look for the "When I receive" hat block. Drag it to the coding area and change it to 3LED, then underneath the block add three broadcasts and edit them as follows.

```
broadcast gpio17on
broadcast gpio27on
broadcast gpi22on
```

This turns every LED on; to turn an LED off we simply swap the **on** for **off**.

Now repeat this structure for 2LED, 1LED and 0LED, remember to be consistent in which LED are turned off. So that's it, save your work and click on the Green flag to play the game.


Connects to all foil using multiple wires
Connects to surgery tool (Tweezers)

Our project is a rather simple circuit that sits inside an A4 document holder, with a lovely picture of a robot stuck upon it.

PROJECT 2 – GPIO ZERO

We covered GPIO Zero in LV23, and since then it has been included as standard in the latest version of Raspbian, so no installation is required. For those using an older version, please refer to **https://pythonhosted.org/gpiozero/#install** for installation instructions.

From the Programming menu open the Python 3 > Idle application. You'll see a Python shell open; click on File > New Window to open a new project window. Before you progress, save the blank document as **Dr-Robot-GPIO-Zero.py** and remember to save your work regularly.

We start the project by importing three classes from the GPIO Zero library, handling the LEDs, the buzzer and our surgery tool, which is classed as an input (button). We also import the **time** library to control our game pace.

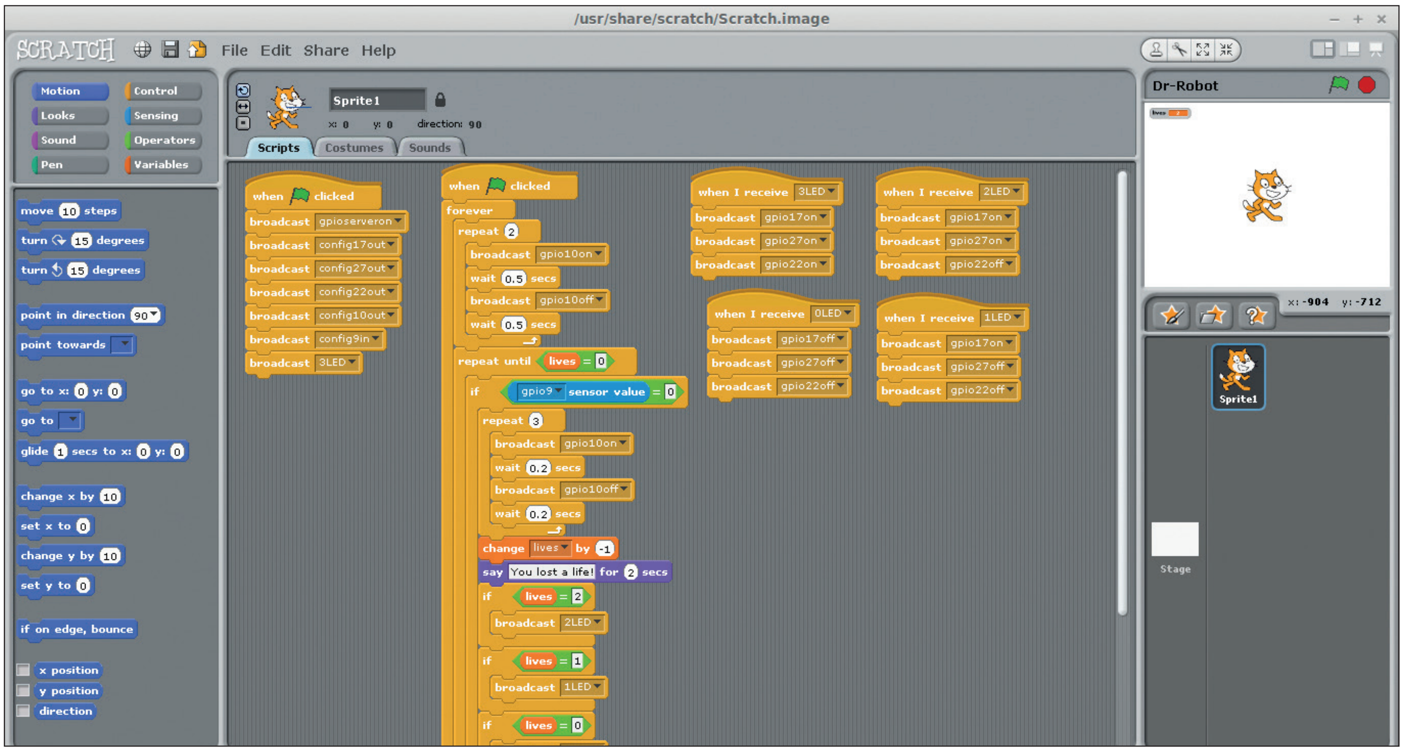
```
from gpiozero import LED
from gpiozero import Button
from gpiozero import Buzzer
import time
```

Next we create five variables that will each store the GPIO pin used for the LEDs, buzzer and surgery tool.

Crafty hacking

Using craft equipment from art and design lessons is a great way for children to incorporate a hack into their curriculum. For example, in science lessons the class could learn about Mars and its inhospitable terrain; in art they could design the terrain of Mars; and in computing they can learn to program a Raspberry Pi robot. This joined up "cross curricular" methodology can help put the classes knowledge into practice and still remain a fun activity.

At the Picademy training events we have a session where the class are shown how to use a motor controller and are then given 30 minutes to build a moving gadget using nothing more than arts and craft materials that can be found in any school. The results of this experiential activity are wide ranging– from simple roulette wheels to interactive goal line technology – but the most important outcome of the session is for learners to try out hacking a project to life and solving problems as they occur.



```
when [flag] clicked
broadcast gpioserveron
broadcast config17out
broadcast config27out
broadcast config22out
broadcast config10out
broadcast config9in
broadcast 3LED

when [flag] clicked
forever
    repeat 2
        broadcast gpio10on
        wait 0.5 secs
        broadcast gpio10off
        wait 0.5 secs
    repeat until lives = 0
        if gpio9 sensor value = 0
            repeat 3
                broadcast gpio10on
                wait 0.2 secs
                broadcast gpio10off
                wait 0.2 secs
            change lives by -1
            say You lost a life! for 2 secs
            if lives = 2
                broadcast 2LED
            if lives = 1
                broadcast 1LED
            if lives = 0

when I receive 3LED
broadcast gpio17on
broadcast gpio27on
broadcast gpio22on

when I receive 2LED
broadcast gpio17on
broadcast gpio27on
broadcast gpio22off

when I receive 0LED
broadcast gpio17off
broadcast gpio27off
broadcast gpio22off

when I receive 1LED
broadcast gpio17on
broadcast gpio27off
broadcast gpio22off
```

Programming the project with the latest version of Scratch is an excellent introduction to coding for beginners.

We call the relevant class LED, Buzzer and Button, and pass the class the GPIO pin as an argument.

```
life1 = LED(17)
life2 = LED(27)
life3 = LED(22)
buzzer = Buzzer(10)
tool = Button(9)
```

Now we create a function to handle turning on multiple LEDs to represent the number of lives we have left. The function is passed an argument, being the number of lives we have. Then an **if..elif** conditional statement is used, the number of lives passed as an argument is compared to hard-coded values, and the correct number of LEDs are lit.

```
def life_counter(lives):
        if lives == 3:
                life1.on()
                life2.on()
                life3.on()
        elif lives == 2:
                life1.on()
                life2.on()
                life3.off()
        elif lives == 1:
                life1.on()
                life2.off()
                life3.off()
        elif lives == 0:
                life1.off()
                life2.off()
                life3.off()
```

Next we create another variable to store the number of lives, in this case three. We then use the variable as an argument and call the function that we have just created.

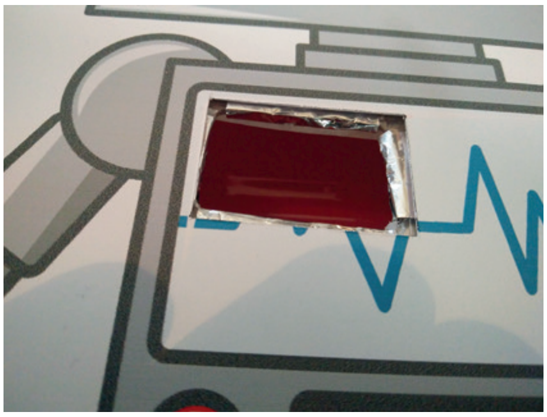
```
lives = 3
life_counter(lives)
```

So now we move into the logic that forms our game. We start by using an infinite loop, **while True**, which will constantly run the code. We introduce a 0.01 second delay to reduce the hit on the CPU and ensure that our code runs smoothly.

```
while True:
        time.sleep(0.01)
```

Still inside the infinite loop we now create a **for** loop, a loop that will iterate a set number of times. In this **for** loop we shall instruct the buzzer to sound, indicating that the game is ready to be played. We turn on the buzzer, then create a 0.5 second delay before turning it off, followed by another delay.

```
for i in range(2):
        buzzer.on()
        time.sleep(0.5)
```


Between the robot picture and plastic case we have aluminium foil connected to the Ground of our Pi. Touching the foil is hazardous to the robot's health.

```
buzzer.off()
time.sleep(0.5)
```

We now come out of the **for** loop and create a new loop inside the infinite loop. Here we use a 'while the number of lives is greater than zero' loop. We evaluate the value of the **lives** variable each time and check that it is greater than 0; if that is the case, the loop repeats. We also introduce a delay to pace our code.

```
while lives > 0:
        time.sleep(0.01)
```

Inside the **while lives > 0** loop we create a condition to check against; this time we are checking to see if the surgeon has touched the aluminium foil, triggering a loss of life. When the foil is touched, the GPIO pin attached to the surgery tool goes from on to off (True to false, 1 to 0) and this is a change of state used to indicate an error.

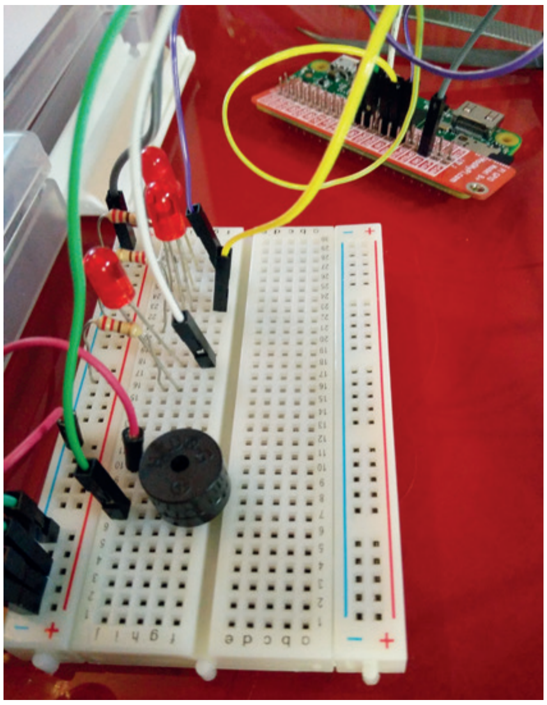
```
if tool.is_pressed:
```

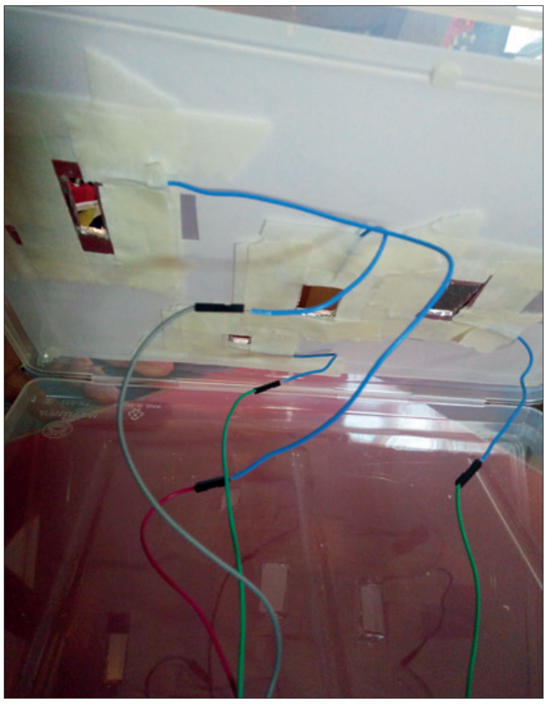
So if the surgeon makes a mistake and touches the metal, the **tool.is_pressed** condition is true and we create a **for** loop that will beep the buzzer three times in quick succession.

```
for i in range(3):
        buzzer.on()
        time.sleep(0.2)
        buzzer.off()
        time.sleep(0.2)
```

Breaking out of the **for** loop we next create a delay before printing that the user has lost a life. We adjust the **lives** variable and one life from its current value. Finally we call the **life_counter** function and pass it the number of lives that the player has left.

```
time.sleep(0.1)
print("You lost a life")
```


A breadboard is an ideal place to build your project. You can quickly test and take apart your project for the optimum layout.


More foil is used for each hole, to create a single circuit for each hole. Each circuit connects to a Ground pin on your Pi, but you could link all the holes together and use just one Ground pin.

```
lives = lives - 1
life_counter(lives)
```

In our final section of code we break out of **if tool.is_pressed** condition and return to the infinite loop. We now have a condition that will activate when the user has no lives left. If that's true, the text 'Game Over' is printed to the screen. A three-second delay takes place before we change the **lives** variable, restoring the three lives that the player receives; this is then indicated by illuminating the LEDS.

```
if lives == 0:
        print("Game Over")
        time.sleep(3)
        lives=3
        life_counter(lives)
```

Ensure that your code is saved and when ready click on Run > Run Module to start the game.

What have we learned?

For this issue we highlighted how one hardware project can be coded using two different methods. Scratch is great at illustrating how the sequence of code works as it is very visual with the design of blocks and loops, and GPIO Zero enables anyone to dip their toe into hardware hacking with Python, which is great for children who want immediate results from their projects.

Coding is a great activity but we should not remain tied to just one language. Once you understand the logic of programming then this knowledge can be transferred to other languages such as Ruby, Perl or JavaScript, and this is key skill for children to grasp in the new Computing curriculum.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

MARIA DB: LEARN THE POWER OF DATA

How to live happy with MariaDB, and reuse its data in ways you didn't know existed.

MARCO FIORETTI

WHY DO THIS?

- *MariaDB* databases probably power websites that you already have. The sooner you learn how they work, the better for you
- Because being able to quickly get every kind of data in or out of a real database is a skill that may save you a lot of work
- *MySQL/MariaDB* knowledge is something employers like…

If you use any Linux desktop, or have any kind of dynamic website, you're using a database. And the more you go on, the more likely it is that you will have to recover those databases from some disaster, or process them in ways that are not possible using only database software.

A database is an archive of data, structured in tables and stored in low-level binary containers. If you could look inside a database, each of its tables would resemble a spreadsheet, with data of the same kind in their separate columns, and different data about the same entity (in database lingo: entry, or record) all in the same row. The "Customers" table of a shop database, for example, may have columns like "Customer code", "Customer name" and "Unpaid orders", and each row (each record), would describe one different customer.

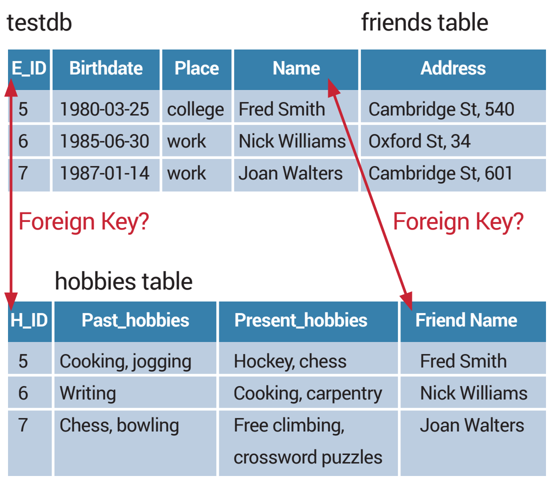
A database is what's known as relational when you can define and use precise relationships among its tables, and the software that manages this is called an RDBMS (Relational DataBase Management System). A column that uniquely identifies each record in a table of a relational database is called its primary key. Relations between tables can be enforced by defining some columns of one table as foreign keys in another. In the shop database above, this would ensure that you cannot enter, in the "Orders" table, a new order from a customer who does not already exist in the "Customers" table.

> A database is an archive of data, structured in tables and stored in low-level binary containers

The standard way to work with relational databases is via commands in an *ad hoc* Structured Query Language, or SQL for short. You may issue those commands indirectly, through a graphical interface, or directly type them inside some *MariaDB* client.

Enter MariaDB

MariaDB (**https://mariadb.org**) is the Open Source, client–server RDBMS most popular on Linux these days. Born as a drop-in replacement for *MySQL*



E_ID	Birthdate	Place	Name	Address
5	1980-03-25	college	Fred Smith	Cambridge St, 540
6	1985-06-30	work	Nick Williams	Oxford St, 34
7	1987-01-14	work	Joan Walters	Cambridge St, 601

H_ID	Past_hobbies	Present_hobbies	Friend Name
5	Cooking, jogging	Hockey, chess	Fred Smith
6	Writing	Cooking, carpentry	Nick Williams
7	Chess, bowling	Free climbing, crossword puzzles	Joan Walters

The logical partitioning of data in tables and, above all, the possibility to define, and enforce, relationships between tables by means of keys, that is related fields that connect them.

MariaDB is relatively easy to use thanks to its programming interfaces in many languages. The client–server architecture means that, in order to create and use any database with *MariaDB*, you must install and run at least two programs. The first of these is the default command line *MariaDB* client, which, somewhat confusingly, is called **mysql**, for compatibility with *MySQL*. The server, which for the same reason is still called **mysqld**, is the part that is always running, and that accesses each database, to execute the requests coming from different clients, which may be on the same or on another computer.

MariaDB commands

Installing the *MariaDB* server and/or client is no problem at all, unless you are using a niche, or experts-only version of Linux. In 2016, all the most popular distributions should offer binary packages for both programs in their standard repositories. You may have to explicitly enable the server to start at each boot, but that's all there is to it.

The SQL commands to create or delete databases, tables and users are all thoroughly explained in the Documentation section of the *MariaDB* Knowledge Base (**https://mariadb.com/kb/en/mariadb/documentation**) or in the latest *MySQL* Reference manual (**https://dev.MySql.com/doc**). We're only going to look at their most basic forms, to give you an idea of how they work and help you understand the manuals (throughout the tutorial, **#>** indicates the Linux prompt; **&** is the *MariaDB* one. Most *MariaDB* output was also removed for brevity):

```
#> mysql
& USE MySql;
& CREATE DATABASE testdb;
& CREATE USER 'linuxvoice'@'localhost' IDENTIFIED
BY PASSWORD 'testpw';
& GRANT ALL ON testdb.* TO 'linuxvoice'@'localhost';
& FLUSH PRIVILEGES;
```

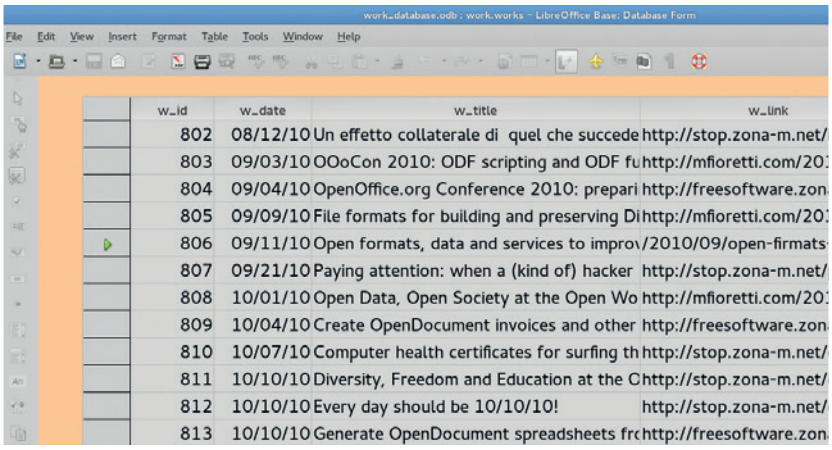
The first line launches the *MariaDB* client. Depending on your *MariaDB* default settings, you may have to run it as root, whenever you want to create or delete databases or users. Remember that *MariaDB* users only exist inside *MariaDB*, but have nothing to do with ordinary Linux accounts, including the administrator one which, like in Linux, is called "root". The five commands above are what you should type at the *MariaDB* prompt in order to:

1 Go into the "main" database of your installation, which stores metadata about all the others.
2 Create a new database, called **testdb.**
3 Add a *MariaDB* user called **linuxvoice**, with password **testpw**.
4 Grant that same user all privileges on all the tables in the **testdb** database. You can create as many databases and users as you want, and then restrict what each user can see or do, with the right options to the **GRANT** command.
5 Tell the *MariaDB* server to flush the old privileges configuration, to load the new one.

Once you have created a database, you can log in to *MariaDB* as a normal user, type the corresponding password, and start creating tables, which you may then fill with data:

```
#> mysql -u linuxvoice -p
& USE testdb;
& CREATE TABLE friends (
 `F_ID` int(11) NOT NULL AUTO_INCREMENT,
 `Birthdate` date NOT NULL DEFAULT '0000-00-00',
 `place` enum('work','high school',
'college','neighborhood') DEFAULT 'neighborhood',
 `name` varchar(255) DEFAULT NULL,
 `address` varchar(255) DEFAULT NULL,
 ...
```

These commands create a table inside **testdb** called **friends**, along with its first five columns, each with a different meaning and a matching *MariaDB* data type: a unique numeric identifier, a date in YYYY-MM-DD format, a predefined set of text constants and finally two strings, to hold each friend's name and address. As above, this is an incomplete snippet of code, only meant to show the look and feel of *MariaDB*.



work_database.odb : work.works – LibreOffice Base: Database Form

w_id	w_date	w_title	w_link
802	08/12/10	Un effetto collaterale di quel che succede	http://stop.zona-m.net/
803	09/03/10	OOoCon 2010: ODF scripting and ODF fu	http://mfioretti.com/20
804	09/04/10	OpenOffice.org Conference 2010: prepari	http://freesoftware.zon
805	09/09/10	File formats for building and preserving Di	http://mfioretti.com/20
806	09/11/10	Open formats, data and services to improv	/2010/09/open-firmats
807	09/21/10	Paying attention: when a (kind of) hacker	http://stop.zona-m.net/
808	10/01/10	Open Data, Open Society at the Open Wo	http://mfioretti.com/20
809	10/04/10	Create OpenDocument invoices and other	http://freesoftware.zon
810	10/07/10	Computer health certificates for surfing th	http://stop.zona-m.net/
811	10/10/10	Diversity, Freedom and Education at the C	http://stop.zona-m.net/
812	10/10/10	Every day should be 10/10/10!	http://stop.zona-m.net/
813	10/10/10	Generate OpenDocument spreadsheets fr	http://freesoftware.zon

You can browse and edit *MariaDB* tables with *LibreOffice*, as in this screenshot, but only raw SQL and the other methods described here can process them automatically.

PRO TIP

You can use everything here to deal with *MySQL* database, for now – be prepared for the future, and keep an eye on the *MariaDB/MySQL* incompatibilities that will surely arrive in the future

Playing with records

If the **friends** table above only had those five columns, you could add a friend named Fred like this:

```
& INSERT INTO friends VALUES(NULL, '1980-03-25',
'college', 'Fred Smith', 'Oxford Street, 21');
```

where the **NULL** value tells *MariaDB* to increment and set the numeric identifier itself. Now uppose Fred moves house? No problem. Assuming that *MariaDB* gave **Fred 5** as **F_ID** value, just update Fred's record:

```
& UPDATE friends set address = 'Cambridge Street, 540'
WHERE F_ID = 5;
```

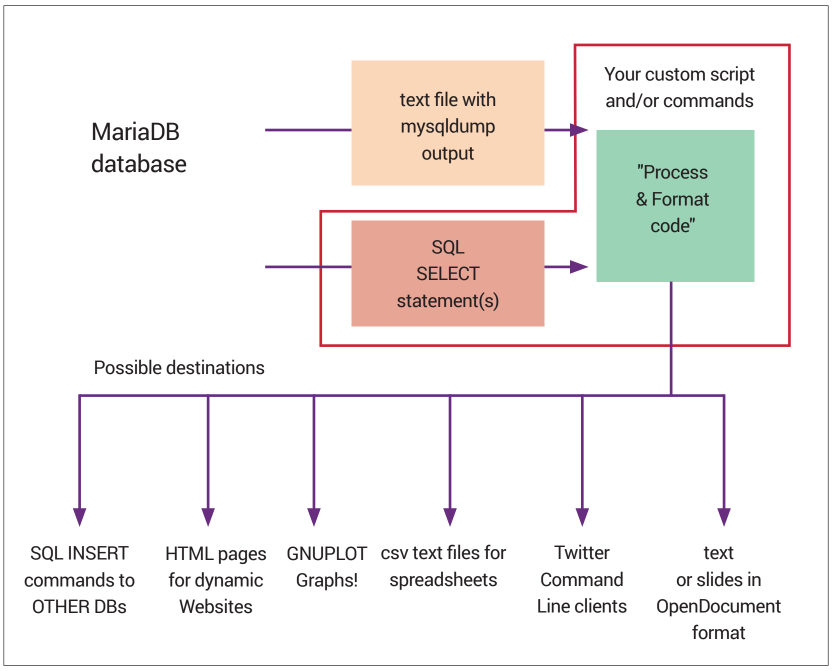
Should you and Fred cease to be friends, you may

MariaDB/MySQL compatibility

The *MariaDB* project was launched six years ago with the goal of becoming "a complete drop-in-replacement for *MySQL*". Some day that aim will be abandoned, but today it still is 100% true between corresponding versions of the two databases. The names of the server and client programs, as well as those of all the configuration and raw, binary files, and of all the common configuration variables, are identical.

The same applies to the SQL syntax, and to the binary interfaces that were originally developed to talk directly to a *MySQL* server from most programming languages. It is still possible to talk to a *MariaDB* server from the *MySQL* client, and vice versa. In spite of all these efforts, really 100% compatibility is not guaranteed. On some Linux distributions, for example, some extra manual work is required to make the *MariaDB* server (but not the *MySQL* one) restart automatically at every reboot. *MariaDB* also has more storage engines available, at least in the official package, than *MySQL*. In a *MariaDB/MySQL* installation, the storage engine is the set of algorithms and low-level libraries that actually create and manage the binary data structures.

There are storage engines optimizsd for speed, others for robustness and many other scenarios. If you used *MySQL* to manage a database with certain non-default storage engines, you may be forced to manually recompile *MariaDB* to migrate that same database. Finally, *MariaDB* has different defaults values for certain variables, and more options than *MySQL*. To know more, read **https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-compatibility**.


MariaDB database
text file with mysqldump output
SQL SELECT statement(s)
Your custom script and/or commands
"Process & Format code"
Possible destinations
SQL INSERT commands to OTHER DBs
HTML pages for dynamic Websites
GNUPLOT Graphs!
csv text files for spreadsheets
Twitter Command Line clients
text or slides in OpenDocument format

From graphs to web pages, when you know the basics of SQL and scripting, it's easy to transform *MariaDB* database, or their textual backups, into almost everything.

remove him from your table:

```
& DELETE FROM friends WHERE F_ID = 5;
```

To list the name and addresses of all your friends who are more than 25 years old, sorted by street, you could ask *MariaDB* to:

```
& SELECT address as A, name as N FROM friends
WHERE Birthdate <= '1990-01-01' ORDER by address;
+----------------------+---------------+
| A                    | N             |
+----------------------+---------------+
| Cambridge Street, 540 | Fred Smith   |
| Cambridge Street, 601 | Joan Walters |
| Oxford Street, 34     | Nick Williams |
```

PRO TIP

Always try a SELECT before an UPDATE or DELETE query, with the same clauses, to be sure that those operations will only alter the records you do want to alter!

The **WHERE** clause is essential. Without it, *MariaDB* would display, delete or alter every record of the selected table!

All SQL statements share the same basic structure shown above: you first define which columns you want to see, how to format each field if necessary, and from which table(s) *MariaDB* should get them. To see only the records that match certain conditions, describe all of them in the **WHERE** clause.

The real power of SQL: joining tables

The **SQL JOIN** operator places two or more tables of a database side by side, to create one temporary, virtual table, that you may filter and display as if it were a normal one. These examples introduce three common types of **JOIN**:

```
& SELECT * FROM friends INNER JOIN hobbies ON
friends.name = hobbies.friend_name WHERE hobbies.
present_hobbies like '%hockey%';
& SELECT * FROM friends LEFT JOIN hobbies ON
friends.name = hobbies.friend_namename;
& SELECT * FROM friends RIGHT JOIN hobbies ON
friends.name = hobbies.friend_name;
```

All commands concatenate rows in such a way that a **friends** row with a certain value goes side by side with the row from **hobbies** in which **friend_name** has that same value. What changes from one **JOIN** example to the other (and remember, these are just three of the available variants) is which rows are considered, and how "holes" in the temporary table are filled.

The temporary table created by the **INNER JOIN** concatenates only the rows of **friends** and **hobbies** that a) have the same value in the **name** and **friend_name** fields (without the **ON** clause, you'd get every row of **friends** concatenated to every row of **hobbies**) and b) a list of **present_hobbies** that includes the string **hockey**.

The second command produces a table that has all the rows from the table on the left of the **JOIN** keyword (that is **friends**) each concatenated with:

- The row with the same **friend_name**, if such a row exists;
- A row with some or all fields set to **NULL** otherwise.

The **RIGHT JOIN** does the same thing, just reversing the tables.

Now, before moving to the next part, let's learn the most important things, that is how to back up your *MariaDB* database with **mysqldump**. Typing this in a Linux shell:

```
#> mysqldump -u linuxvoice -p --extended-insert=false
friends> friends-backup.sql
```

would save all the SQL instructions to rebuild from scratch all the content of your **friends** database in the text file **friends-backup.sql**, when loaded from the *MariaDB* prompt:

```
#> mysql -u linuxvoice -p friends
& source friends-backup.sql;
```

Why bother with all this?

There are lots of programming languages with *MariaDB*/*MySQL* interfaces. Raw SQL commands are plain text strings, and the whole content of a *MariaDB* database can be dumped into one file, as a sequence of plain text commands, each with a well defined structure. Taking all that into account, it's easy to realise that creation, recovery and reuse of *MariaSQL* databases, are much easier than you may suspect.

Because the commands are plain text, we don't need to use SQL to manipulate them; we could, for example, dump the whole content of the database, or of a single table, into a flat text file, one record per line, whith **MySqldump**. At that point, it will be easy to cut, slice, combine, alter and reformat that text with tools like *sed*, *awk* or *grep*, inside shell scripts.

The other method consists of querying the database from inside a script, using the *MariaDB/MySQL* libraries for the chosen language, and playing with the results. Listing 1 shows some lines of Perl that detect, format and display, more flexibly than SQL could do, the duplicate entries in a budget database:

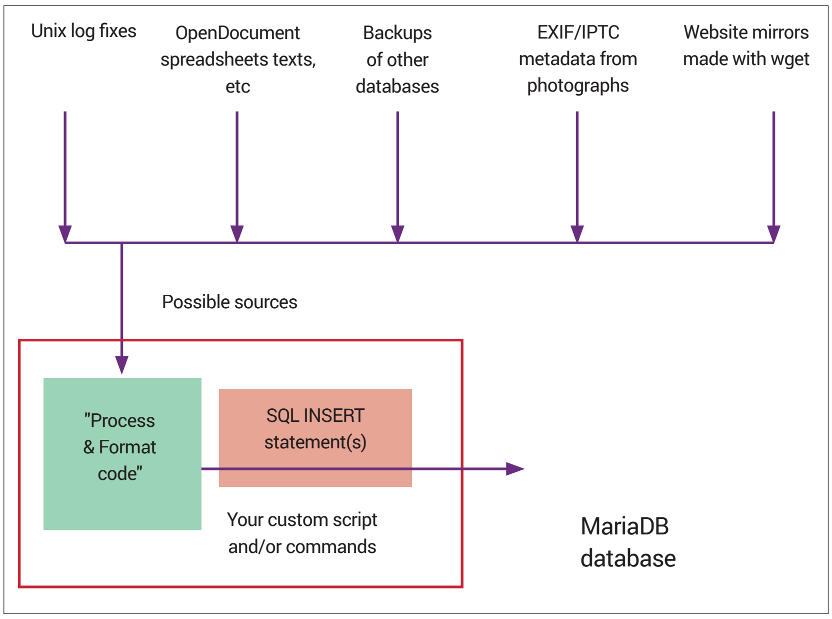
```
Listing 1
   1      use DBI;
```

```
2    use DBD::MySql;
4    my $database = "budget";
5    my $host     = "localhost";
6    my $port     = "3306";
7    my $user     = "budget_user";
8    my $pw       = "budget_password";
9    my $dsn      = "dbi:MySql:$database:localhost:3306";
10    my $DB      = DBI->connect($dsn, $user, $pw);
12    my $query = qq~select b_date, b_item, b_amount, b_id from budget_list where b_date >= '2006-01-01' order by b_date;~;
13    my $query_handle = $DB->prepare($query);
14    $query_handle->execute();
15    $query_handle->bind_columns(\$b_date, \$b_item, \$b_amount, \$b_id);
17    while($query_handle->fetch()) {
19     if ($DOUBLE_ITEMS{"$b_date|$b_amount"}{'exists'} eq 'Y') {
20         print $DOUBLE_ITEMS{"$b_date|$b_amount"}{'value'};
21         printf "    %10.10s  : %-40.40s   %8.2f  %-8s\n",
22                         $b_date, $b_item, $b_amount, $b_id;
23       } else {
24         $DOUBLE_ITEMS{"$b_date|$b_amount"}{'exists'} = 'Y';
25         $DOUBLE_ITEMS{"$b_date|$b_amount"}{'value'} =
26           sprintf "ORIG:   %10.10s  : %-40.40s   %8.2f %-8s\n",
27           $b_date, $b_item, $b_amount, $b_id;
28       }
29     }
```

The first ten 10 of Listing 1 show one way to load Perl modules that talk to *MariaDB*, initialising all the necessary variables and using them to connect to a database called **budget**. The next four statements prepare a query, execute it and bind the columns it returns to Perl variables with the same names. In line 17, the script loops through each row returned by the query. If the **%DOUBLE_ITEMS** array already contains an entry with the same combination of **date** and **amount**, both that and the current record are printed, as possible duplicates. Otherwise, the current record is reformatted, and inserted into **%DOUBLE_ITEMS**.

With the same base technique you may quickly write code (in Perl or many other languages!) that, in the same run, connects to different *MariaDB* databases (or other data sources!), processes all their contents as you wish, and prints out the results as you want. More detailed, useful outputs that a script like that may generate include:

- Actual SQL statements that you may execute from the same script, with the same connection technique, or print out to a text file.
- A CSV (Comma Separated Values) text version, directly usable in spreadsheets like *Calc* or *Excel*, of any data generated from processing the same records


Unix log fixes
OpenDocument spreadsheets texts, etc
Backups of other databases
EXIF/IPTC metadata from photographs
Website mirrors made with wget
Possible sources
"Process & Format code"
SQL INSERT statement(s)
Your custom script and/or commands
MariaDB database

A few lines of SQL commands and glue code can load everything, from photo metadata to system logs, into a *MariaDB* database, for further analysis.

Even the inverse passage from CSV to database is simple, once you know the basic trick. These few lines of code, again in Perl:

```
while (<>) {
chomp;
($b_date, $b_item, b_amount) = split/\s*,\s*/;
# process date, item and amount as you wish!
print "INSERT INTO budget_list VALUES(NULL,'$b_date', '$b_item','$b_amount');\n";
}
```

are (almost) all you'd need to read a CSV file, one line at a time, load its fields in Perl variables, process them as needed, and generate the valid SQL statements that would insert them in a *MariaDB* database.

Last but not least, websites. Many dynamic websites are powered by *MariaDB* or *MySQL*. When we had to resurrect an old *Drupal* website, for example, a quick look at its **mysqldump** backups showed that these three Perl statements:

```
$PRINT = 'y' if ($_ =~ m/^CREATE TABLE `node`/);
$PRINT = 'n' if ($_ =~ m/^CREATE TABLE `performance_detail`/);
print "$_" if ('y' eq $PRINT);
```

were all we needed to print only the SQL commands containing the fields with the full, actual text, and nothing more, of all its pages. After that, less than one hour of trial and error later we had other 50 lines of code that gave us a folder with all those web pages, each in a separate file named after their title and publication date, ready for reuse. The gory details and full source code are available online at **http://freesoftware.zona-m.net/?p=124**. Take it from us: knowing how to talk to *MariaDB* and reuse its data can save you lots of time, and be a lot of fun.

PRO TIP

The *MariaDB* command line client keeps a history of its commands, just like the Unix shells. Use it!

Marco Fioretti is a campaigner and writer on issues surrounding free software, ethics and the environment.

ILLUMINATE YOUR LIFE WITH LINUX PART 1

Take your first steps in home automation by using Linux to control your lights.

MARK CRUTCH

WHY DO THIS?

- Gain control over your lights
- Get on the IoT bandwagon
- Party like an old school disco

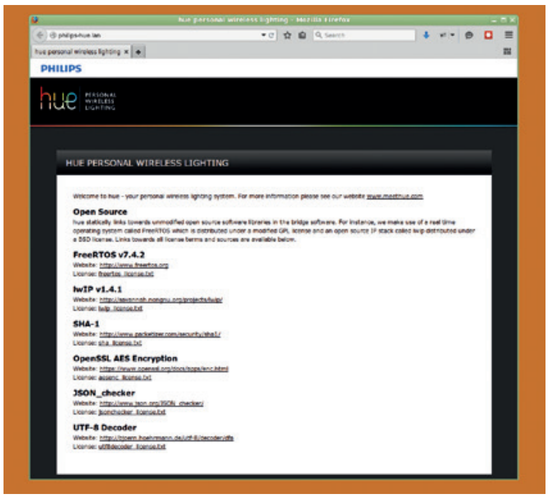
We're fed up with the term "Internet of Things". Yes, a new wave of low-power devices and ubiquitous wireless connections mean that ever more devices are able to report their status to the world. However, we've been using Arduinos and Raspberry Pis to add "intelligence" to dumb devices for long enough that this isn't really anything new. To be quite frank, we're very much looking forward to the day when internet connectedness comes as standard, and we can go back to just using the term "things".

As jaded as we are by the terminology, we still feel a thrill every time we're able to control a physical device with just a few lines of code. It's an empowering skill – being able to link the wider world of data and messages with changes in your physical environment. Perhaps there's no easier way to get a feel for that power than to control your own domestic lights.

We opted to use the Philips Hue range of "smart" bulbs. These sell for a ridiculous £50 per bulb, or a slightly more reasonable £15 for the Lux range – white bulbs that can be dimmed but can't change colour. You also need a "bridge" to control the lights, which can only be purchased as part of a Starter Kit, setting you back £150 just to get going (three Hue bulbs and a bridge), or £50 for the Lux option (two bulbs and a bridge). You can use Hue bulbs with the bridge from a Lux kit, and vice versa, so you can at least start off with a Lux kit and add colour later.

> The bulbs form a mesh network, which makes it easier to extend your lighting system around the house

The system uses a low-power wireless protocol called ZigBee. Although it's wireless, it's not the same as Wi-Fi – hence the need for the bridge, which connects the lights' ZigBee network to a wired Ethernet cable that plugs into your router or switch. This enables any device on your network to send data to the bridge, which then routes it to the lights. The bulbs themselves form a mesh network, which makes it easy to extend your lighting system around the house, even to those corners that Wi-Fi never reaches.


hue personal wireless lighting - Mozilla Firefox
philips-hue.lan
Search
hue personal wireless lighting
PHILIPS
hue PERSONAL WIRELESS LIGHTING

HUE PERSONAL WIRELESS LIGHTING

Welcome to hue - your personal wireless lighting system. For more information please see our website www.meethue.com

Open Source

hue statically links towards unmodified open source software libraries in the bridge software. For instance, we make use of a real time operating system called FreeRTOS which is distributed under a modified GPL license and an open source IP stack called lwip distributed under a BSD license. Links towards all license terms and sources are available below.

FreeRTOS v7.4.2
Website: http://www.freertos.org
License: freertos_license.txt

lwIP v1.4.1
Website: http://savannah.nongnu.org/projects/lwip/
License: lwip_license.txt

SHA-1
Website: http://www.packetizer.com/security/sha1/
License: sha_license.txt

OpenSSL AES Encryption
Website: https://www.openssl.org/docs/apps/enc.html
License: aesenc_license.txt

JSON_checker
Website: http://www.json.org/JSON_checker/
License: jsonchecker_license.txt

UTF-8 Decoder
Website: http://bjoern.hoehrmann.de/utf-8/decoder/dfa
License: utf8decoder_license.txt

The default bridge homepage lists the open source projects that it's based on.

To control the lights from a Linux box, we'll use the RESTful API exposed by the bridge, which requires us to find the bridge's IP address or hostname. Ours exposed itself via Zeroconf as **philips-hue.lan**, but you can probably get its IP address from your router's web interface. The bridge also communicates back to Philips, telling them its local IP address. This does mean that you can find it by browsing to **https://www.meethue.com/api/nupnp** from within your network, but we'd still prefer it if our hardware didn't talk to an external server behind our backs! Pointing a web browser at either the IP address or the Zeroconf hostname displays the default web page for the bridge, which just shows some details about the open source components it uses.

Let there be light!

Although it's refreshing to see a product so openly presenting this information, we can't help feeling that the bridge could benefit from serving a password-protected administration interface to let you easily turn lights on and off from a browser. What it does have, however, is a debugging screen that lets you submit commands to the bridge and view its responses. It's not linked to from the main page, so you'll have to modify the URL in the browser to visit **http://philips-hue.lan/debug/clip.html** – replacing

philips-hue.lan with the bridge's IP address, if necessary. You should see the "CLIP API Debugger", which we'll simply refer to as the CLIP screen from now on.

Most things in the API are restricted to registered users to prevent a malicious program from playing havoc with your lights, so we need to create a user account on the bridge. New users can only be created in a 30-second window after pressing the hardware button on the bridge, so we'll prepare the data we want to send first, then press the button, and then finally send the request. The address to enter into the "URL" field of the CLIP screen couldn't be simpler:

```
http://philips-hue.lan/api/
```

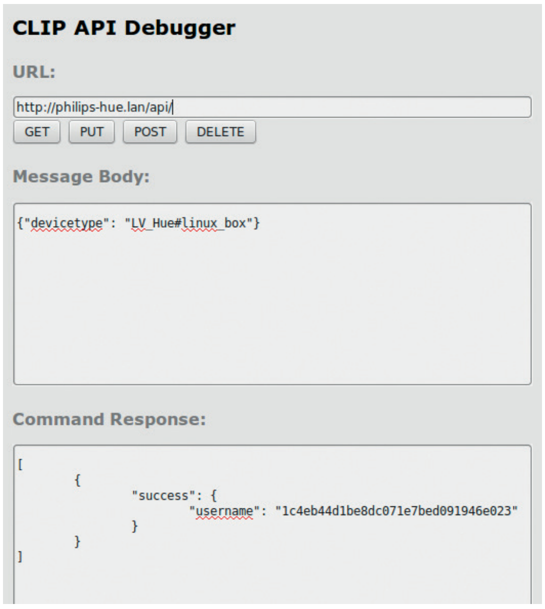
The API requires a **devicetype** parameter, which consists of a pair of strings to identify the name of your application, and the name of the device you're using it on. For our simple test purposes the values aren't terribly important, so we'll use **LV_Hue** as the application name, and **linux_box** as the device. These are separated by a hash character, then wrapped as a JSON object, resulting in the following string that has to be entered in the Message Body box:

```
{"devicetype": "LV_Hue#linux_box"}
```

Now's the time to put on your running shoes, enlist the help of a glamorous assistant, or just move your laptop closer to the bridge; firmly press the button in the middle of the bridge then, within 30 seconds, click on the "POST" button in the CLIP screen. You should see the Command Response panel fill with something similar to this:

```
[
 {
  "success": {
    "username": "1c4eb44d1be8dc071e7bed091946e023"
  }
 }
]
```

Both this and the parameter you sent in the Message Body are encoded in JSON format. This is a simple serialisation of JavaScript data structures, where items in curly braces represent objects (collections of **name:value** pairs), and items in square brackets represent arrays. In this case we sent an object with a single property called **devicetype** whose value was **LV_Hue#linux_box**. We received an array



```
http://philips-hue.lan/api/
```

```
{"devicetype": "LV_Hue#linux_box"}
```

```
[
        {
                "success": {
                        "username": "1c4eb44d1be8dc071e7bed091946e023"
                }
        }
]
```

The CLIP screen lets you send arbitrary data packets to the bridge for testing and debugging.

which contains just a single entry: an object with a property called **success** whose value is another object with a property called **username**. The value of that property is the long hexadecimal string, which is the real payload we need. That string will form part of every other call we make, so that the bridge will accept our instructions as coming from an authorised application.

Put some colour in your life

From now on you'll need to not only replace the hostname in our code, but also swap the hexadecimal string for the one that your own bridge returned. For the sake of space we've abbreviated ours to **1c4...023**, but you should use your full value. Let's start by getting a list of all the lights that are connected to the bridge. Enter your version of this URL into the CLIP screen:

```
http://philips-hue.lan/api/1c4...023/lights
```

Then click the GET button. You should receive a lengthy response detailing the state and capabilities of the lights that your bridge knows about. Each light is represented by a numbered property in the top-level JSON object, and the value of each property is an object, which descends even further into more

PRO TIP

You can use the CLIP screen to test any of the Hue's APIs without writing a single line of code.

Why choose Hue?

There are plenty of manufacturers selling allegedly "smart" bulbs. At the cheap end of the market you can find colour-changing bulbs that are controlled via a dedicated infrared remote control. Slightly more expensive are similar bulbs with Bluetooth controls. These are intended to appeal to smartphone users, but rarely offer an open or documented API, and suffer from range limitations.

We prefer something more sophisticated which is designed to work as part of a larger network. Two contenders stand out: the Philips Hue/Lux range that we've used in this tutorial, and LIFX bulbs (**www.lifx.com**), which started out as a Kickstarter project but are now available for general purchase. In both cases the price per bulb is similar, but the LIFX bulbs are connected via Wi-Fi, with no need for a bridge. You can therefore get started with a single LIFX bulb, compared with Philips' requirement for you to buy a whole starter kit.

Although it's more expensive to get started, there are some advantages to the Hue bulbs, which led to us choosing them. The use of a bridge means that scheduled lighting changes continue to occur even if our network is down. The mesh network formed by the bulbs means that we have no problem with signal propagation, even in areas that have no Wi-Fi signal. Finally the bridge offers a simple RESTful API that's available locally. LIFX does have a similar API, but only via its own servers; if your internet connection is down, or their servers are unavailable, you can't use it to control the lights. There is a UDP-based protocol that can be used for local applications, but it's not so developer-friendly.

properties and objects. You can request the details for a single light by appending its number to the end of your URL (eg **http://philips-hue.lan/api/1c4...023/lights/1**) and sending a GET request. To switch the light on, we need to PUT a vaue of **true** into the **on** property of the **state** object. Enter the following two lines into the URL and Message Body fields, respectively, then press the PUT button.

```
http://philips-hue.lan/api/1c4e...023/
lights/1/state
{"on":true}
```

Note that there are no quotes around the value, as this is a JavaScript Boolean, not a string. To turn the light off again, change the value to **false**, then press the PUT button again. If the light's too bright, try this payload to both turn it on and set it to half brightess (on a scale of 1 to 254):

```
{"on":true, "bri":127}
```

The CLIP screen is great for experimenting with the API, but we really want to put the power of our lights to practical use. Because all we're doing is sending HTTP requests, you can control the lights from just about any programming language, or even directly from the command line. Create the following *Bash* script as **hue_alert.sh**, changing the first three variables to reflect your own setup and the number of the light you want to affect.

```
#!/bin/bash
HOST=philips-hue.lan
USER=1c4eb44d1be8dc071e7bed091946e023
LIGHT=1
API=lights/$LIGHT/state
PAYLOAD='{"alert":"select"}'
URL=http://$HOST/api/$USER/$API
curl -X PUT -d $PAYLOAD $URL
sleep 1
curl -X PUT -d $PAYLOAD $URL
sleep 1
curl -X PUT -d $PAYLOAD $URL
```

Make it executable using **chmod u+x hue_alert.sh**. You'll also need to install **curl** from your package manager if it's not already on your system. Running this command – using **./hue_alert.sh** from the directory it's saved in – should flash your light three times. Where it really becomes useful, however, is in a situation like this:

```
./long_running_script.sh && ./hue_alert.sh
```

Start your long-running script like this and you can head down to the living room to watch TV knowing that the lamp in the corner will flash when your script has completed. Or how about using the venerable **at** command to flash the lights at a particular time. Need a reminder to do something in 20 minutes?

```
at now+20min
at> ./hue_alert.sh
at> ^D
```

Enter the first line and press the Return key. Then type in a series of commands to execute, one per line. This example just flashes the lights, but you might choose to do something else as well. Finally, press Control+D to end. The **at** command can take a variety of different time definitions, allowing you to execute the command at a specific time, or to include a relative offset – which is what we've done here by setting the specific time to **now**, then adding a 20 minute offset to it.

Controlling your lights from the command line has its uses, but we want to be able to trigger changes from a variety of different inputs. For that kind of task, we prefer to use a language such as Python. To make the code simpler we'll use the **requests** library, so the first step is to install that:

```
pip install requests
```

Now we can write a simple Python program to turn a light on at half brightness. Save the following as **hue_light_on.py**:

```
#!/usr/bin/env python
import requests
host = 'philips-hue.lan'
user = '1c4eb44d1be8dc071e7bed091946e023'
light = 1
payload = '{"on":true, "bri":127}'
api = 'lights/{}/state'.format(light)
url = 'http://{}/api/{}/{}'.format(host, user, api)
r = requests.put(url, payload)
```

You can run this using **python hue_light_on.py**, or by making the file executable as we did with the shell script previously. If you're having trouble getting the code to work, and want to see the error messages coming back from the bridge, add the following line to the end of the code.

```
print r.text
```

You'll notice a lot of similarity between the *Bash* script and the Python version. It makes sense to split the **host**, **user** and **light** values out into variables at the top of the file, to make it easier to reuse the code. After that, it's just a case of compiling the parts together to

PRO TIP

Check the dimensions of the bulbs before ordering – not all light fittings will take them, even if the connector is the right type.

PRO TIP

Make sure you order the correct lights for the fittings you have. You can buy adaptors but they increase the overall length of the bulb.

Accessing the complete API

The Hue API is documented at **www.developers.meethue.com**. There is a "Getting Started" section which describes enough of the API to let you create a user on the bridge and change the state of your lights. More extensive APIs, to control groups of lights or whole lighting scenes are available if you create a free account on the site. This requires you to agree to some terms and conditions, which won't affect you when developing normal applications, but it does prevent anyone legitimately creating a bridge emulator or proxy using the documented APIs on the site. A proxy would greatly open up the possibilities of the system, allowing commands to be intercepted and modified, or offering additional input and control options. It would also allow Hue apps to control non-Philips lights, which probably explains the restriction, but we reckon Philips would actually sell even more lights if they make it easier to interface them with other software and hardware.

Despite this restriction, there are a couple of Hue emulators available on the internet, so a Pandora's box of bulbs has already been opened. Come on Philips, remove this silly restriction and get behind the creativity of your developer community!

make a simple HTTP request. It is important to use the correct request type, though, as the bridge will just return an error message if you try to send a GET request to an API that expects a POST, or a PUT to one that expects a DELETE.

So far we've used commands that work with either Hue or Lux lights, but if you've spent the extra money on a Hue bulb you probably want to know how to get some colour out of it. Let's start by giving you a bright blue light. Modify the payload line to this:

```
payload = '{"on":true, "hue":46920, "sat":254, "bri":254}'
```

Run the script and you should find that your light changes to blue. Obviously we've turned it on and set it to full brightness, but what of the **hue** and **sat** values? The former sets the basic colour of the light on a scale from 0 to 65535. The scale wraps round, so that both ends are red, green is at 25500, and blue is the value of 46920 that we've used in our script. Try modifying it to turn the light green or red – then try other values to find oranges, purples and more. The **sat** value, refers to the **saturation** of the colour: 254 gets you 100% colour, while lower values result in less colour and more white. At a value of 0 you'll get a completely white light.

Shades of light and dark

Sometimes there's a need to increase or decrease the brightness of a bulb by a relative amount, rather than set it to an absolute value. We could do this in code using the parameters we've looked at so far, but there's also a **bri_inc** property that simplifies the process. Despite the name, it can also be used to decrease the brightness by passing a negative number. This payload will nudge the brightness down:

```
payload = '{"on":true, "bri_inc":-30}'
```

There are equivalent parameters for adjusting the hue (**hue_inc**) or saturation (**sat_inc**). The **hue_inc** parameter wraps its result, so adding 45,000 to a red hue will give you something blue, whether your red value is 10 or already up at 65,000.

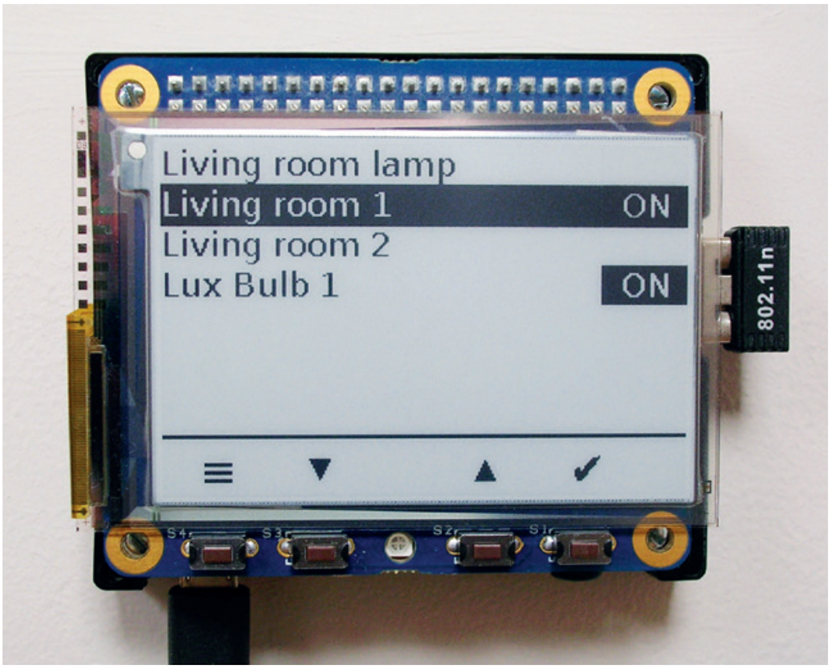
With our **hue_inc** parameter and a little randomness it's easy to simulate some disco lights. Save this as **hue_disco.py** and execute it as usual:

```
#!/usr/bin/env python
import requests
import random
import time
host = 'philips-hue.lan'
user = '1c4eb44d1be8dc071e7bed091946e023'
light = 1
payload = '{"on":true, "hue":0, "sat":254, "bri":254}'
api = 'lights/{}/state'.format(light)
url = 'http://{}/api/{}/{}'.format(host, user, api)
while True:
 r = requests.put(url, payload)
 payload = '{{"hue_inc":{}}}'.format(random.
randint(0,65000))
 time.sleep(random.random() + 0.1)
```

Philips recommends that a light shouldn't be changed more than 10 times a second, hence the addition of 0.1 to whatever random number is used for the **sleep()** call. As we've used an infinite loop for the main part of the code, you'll need to press Ctrl+C to quit. We'll leave it up to you to put in nicer keyboard handling or extend it to multiple lights.

The snippets of code we've presented here are only intended as the basic building blocks of something more comprehensive. A real application should find the bridge and register a new user ID automatically the first time it is run. It should also query the bridge for information about the lights, to determine how many are available, as well as their capabilities. Each light has a "friendly" name that should be presented to the user, rather than just referring to them by number.

But don't let all that put you off throwing together a few little scripts for your own use, with hard-coded user IDs and bridge addresses. The Internet of Things is all about getting disparate devices to talk to each other to make our lives a little better. It's about lights that change colour when you're mentioned in a tweet, or that turn off automatically when your phone is taken out of range of your Wi-Fi access point. We're a long way from cross-device protocols that enable such capabilities with ease, so until then we may have little choice but to hack things together with some less than perfect shell scripts. Which suits us fine, because that's the fun bit.



We used a Raspberry Pi A+, an E-Ink screen (**www.percheron-electronics.uk**) and some custom code based on the Python in this article to create our own wall-mounted lighting controller.

> You can watch TV knowing that the lamp in the corner will flash when your script has completed

Mark Crutch has recently cancelled his holiday to Iceland, now that he can recreate the aurora borealis in his living room, any time he wants.

SED: BUILD A WEB FRAMEWORK

Who needs Apache and PHP when you've got Bash and Sed?

BEN EVERARD

WHY DO THIS?

- Get a better understanding of text processing on Linux
- Learn how the web works

Sed, the Stream Editor, is one of the tools that you find on almost every Linux system. Its basic purpose is to edit a piece of text based on a set of rules that are usually built around regular expressions (often shortened to the snappier 'regexes'). In this tutorial, we're going to abuse *Sed* to build a web framework that can serve up HTML pages and generate dynamic content, but before we dive into the web side of things, let's start by taking a look at *Sed* itself.

Sed works by taking input either from a file or standard input, applying a set of transformations, then outputting the result. The format for this is:

```
sed <program> <file>
```

The program is written in the *Sed* programming language, and if **<file>** is omitted, *Sed* takes its input from Standard Input (Stdin). *Sed* is a highly specialised programming language designed just to apply transformations to text; however, it is Turing complete, which is a computer science way of saying that anything that can be programmed in another language can also be programmed using *Sed*.

The standard first program with any language is 'hello world', but this poses us a little bit of a problem in *Sed* because, unlike most languages, we can't just issue a simple print statement. We have to write a program to manipulate the lines until we get as far as 'hello world'.

Let's take a look at one option:

```
echo "hello" | sed "s/$/ world/"
```

Here, we pipe the text **hello** into the *Sed* program **"s/$/ world/"**. The **s** character means search and replace. Following the **s** we need two arguments that are separated by forward slashes – the first is the regular expression to search for and the second is the text to substitute for this expression. In this particular case, the **$** character matches the end of a line, so this program simply adds **<space>world** to the end of a line. If you run this command, it will output **hello world**, but it's not a true 'hello world' because you have to pipe in **hello** in order for it to work.

Primitive, yet functional

A slightly more advanced option is as follows:

```
echo "anything" | sed "s/.*/hello world/"
```

This uses the regular expression **.***, where the dot character matches any single character and the asterisk following it tells *Sed* to match that character zero more times. In other words, **.*** matches anything at all including nothing. In this case, *Sed* has to decide what to match **.*** to. It could match the first character, or any number of characters on the line. When it has a choice like this, *Sed* will always match the highest number of characters possible; therefore **.*** will always match the entire line.

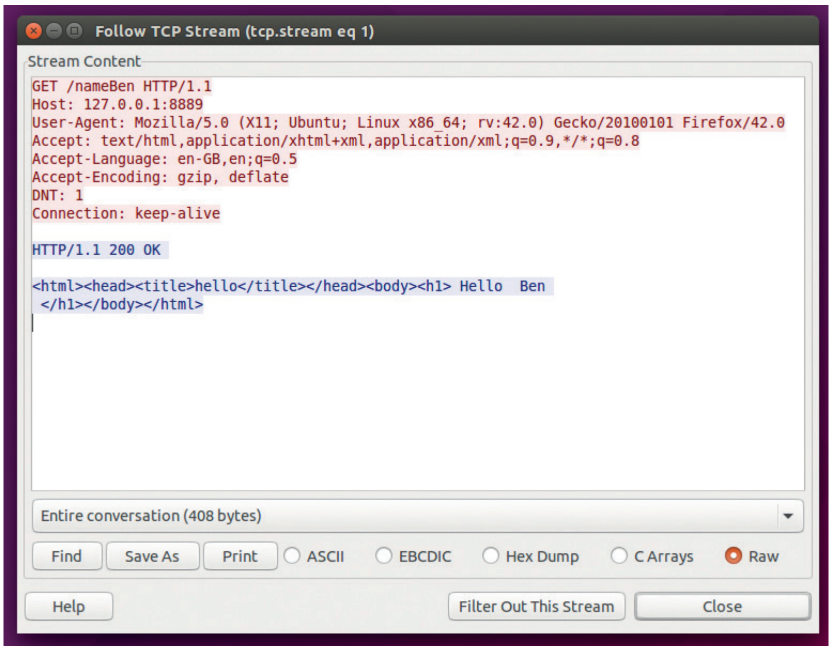
This is a little better than our first example, but it's still not a proper hello world program because it will output **hello world** once for every line that's sent into the program. For example, if you run the following, you'll get a screen full of **hello world**s:

```
dmesg | sed "s/.*/hello world/"
```

The **dmesg** command outputs all the messages from the Linux kernel. It can be useful in diagnosing hardware problems, but here we're just using it to output large amounts of text.

We can make our 'hello world' program a little better by chaining commands. A semicolon can be used to join more than one *Sed* command together, and they will then run one after another. The **q** command quits *Sed*, so we can match, then replace the first line and then quit with the following:

Wireshark's 'follow TCP stream' feature is a great way to get to know text-based protocols such as HTTP.



```
GET /nameBen HTTP/1.1
Host: 127.0.0.1:8889
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:42.0) Gecko/20100101 Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive

HTTP/1.1 200 OK

<html><head><title>hello</title></head><body><h1> Hello  Ben
 </h1></body></html>
```

```
dmesg | sed "s/.*/hello world/;q"
```

Another approach that gets the same result is to use *Sed*'s range function to only match the first line. The problem with this that even if you only tell *Sed* to change the first line, it will still print every other line. To get around this we need two things: the command line argument **-n**, which tells *Sed* not to print out every line by default; and the **p** argument to **s**, which tells *Sed* to explicitly print any lines that have been changed.

```
dmesg | sed -n "1,1 s/.*/hello world/p"
```

The range of lines that we want to search comes before the main command. In this case, we've used line numbers, so this matches just the first line (1,1). Another way of using ranges in *Sed* is to enter two regular expressions separated by a comma. When you do this, *Sed* will start processing at the first instance of the first regular expression and finish at the last instance of the last regular expression.

That's enough about *Sed* to get us started. Let's see how to misuse this editor to make a web server.

Built to serve

Our *Sed* server needs a way of attaching to a TCP port so that anything that's sent into the port gets sent to standard input, and anything that *Sed* outputs gets sent through the TCP stream. If we were building a server to be deployed across many machines that needed to be managed, we'd use the init process for this. Both *init.d* scripts and *Systemd* services can handle this easily. However, since we're just using our server for a bit of fun, there's no need to go to this effort. The *Netcat* (**nc**) tool offers us the functionality we need, so we just need a wrapper script to attach our server to the port:

```
#!/bin/bash
while true ; do nc.traditional -l -p 8889 -e 'run.sh'; done
```

There's more than one version of **nc** and not all of them understand the **-e** flag, which attaches the request to an external process. On Ubuntu-based systems, you'll need to install the **netcat-traditional** package and use the command **nc.traditional** as we have done here. On other distros, you may be able to use the regular **nc** command.

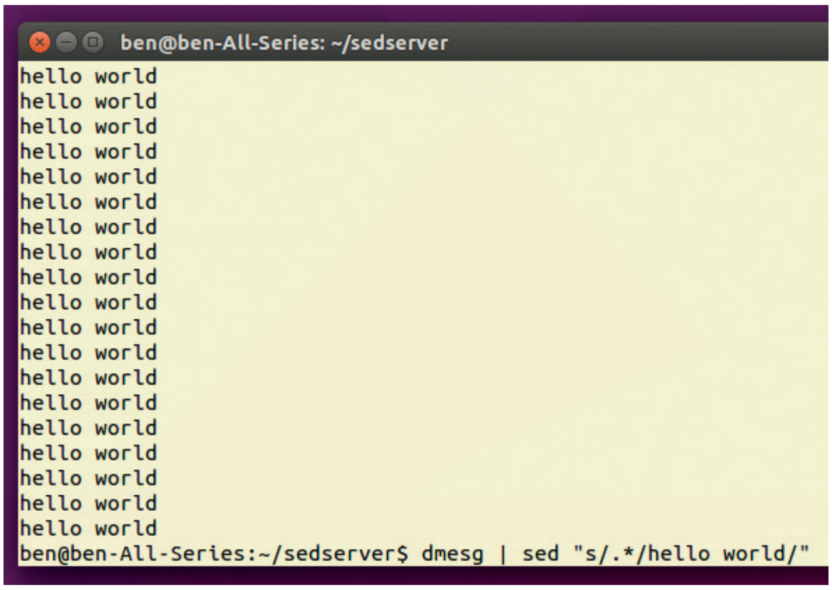
All this script does is run an infinite loop passing each new request onto the **run.sh** script (which will contain our *Sed* program). We've used port 8889 here, but it will work equally well on any port that's not currently in use.

We now just need to create a *Sed* script that takes incoming HTTP requests and replies with the data we want. To do this, we need to know what these HTTP requests will look like, so let's take a look at the requests that are coming in. We can grab the HTTP request by creating **run.sh** and entering the following:

```
#!/bin/bash
cat >> httpoutput
```

Save this as **run.sh**, save the first *Netcat* script as **server.sh** and make them both executable with **chmod +x *.sh**. Then start our web server with **./server.sh** and point your browser to **localhost:8889**. This won't actually load a page (since our server is only saving the contents of the request). Once the page is trying to load, take a look in **httpoutput** and you should see something like:

```
GET / HTTP/1.1
Host: localhost:8889
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:42.0) Gecko/20100101 Firefox/42.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
```



```
ben@ben-All-Series: ~/sedserver
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
ben@ben-All-Series:~/sedserver$ dmesg | sed "s/.*/hello world/"
```

The *sed* command at the bottom replaces every input line with "hello world"

A semicolon can be used to join more than one Sed command together, and they will then run one after another

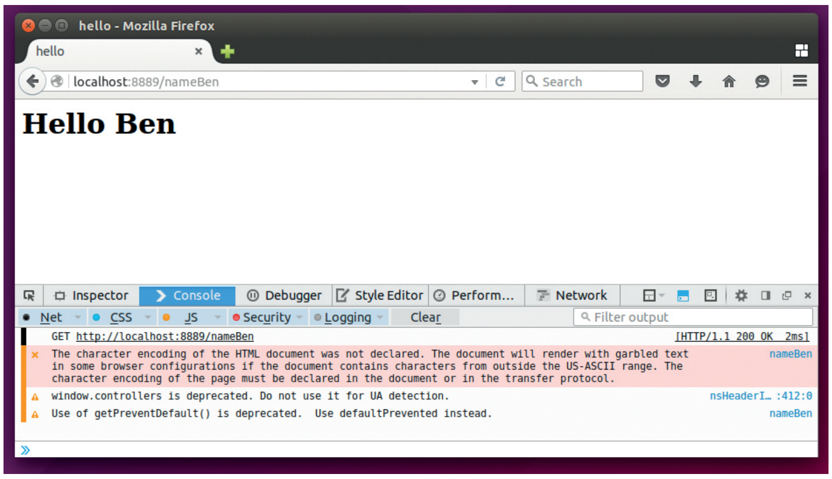
Regular expressions

Much of *Sed*'s power comes from regular expressions (or regex). This allows us a very powerful method of defining what text we want to match our *Sed* rules. Regular expressions define a pattern against which a given piece of text is matched. Here, we'll give a really quick overview, but for a more detailed look at regular expressions, see the *Grep* tutorial from Linux Voice issue 1, which is available at **www.linuxvoice.com/issues/001/bash.pdf**.

Any letter matches itself (the matches are case sensitive), but many symbols have special meanings. A dot matches any single character; **^** matches the start of a line; **$** matches the end of a line; and ***** means 'match the previous character zero or more times'.

Here are a few examples:

- **^hello** will match any line beginning with **hello**.
- **^hello$** will match a line that contains only **hello**.
- **^a.*b$** will match any line that begins with an **a** and ends with a **b** regardless of the characters in between.
- **^...$** will match any line containing exactly three characters.



```
GET http://localhost:8889/nameBen                                [HTTP/1.1 200 OK  2ms]
The character encoding of the HTML document was not declared. The document will render with garbled text    nameBen
in some browser configurations if the document contains characters from outside the US-ASCII range. The
character encoding of the page must be declared in the document or in the transfer protocol.
window.controllers is deprecated. Do not use it for UA detection.                 nsHeaderI… :412:0
Use of getPreventDefault() is deprecated.  Use defaultPrevented instead.          nameBen
```

The developer tools in *Firefox* will give you details of the HTTP requests sent and the responses received.

```
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Cache-Control: max-age=0
```

This is an HTTP request. It contains all sorts of useful information for a web server, and fully featured web servers will use the different parts in different ways. Our server isn't going to speak the full HTTP protocol, but just enough of it to allow a web browser to successfully request a page. For this, we'll use a single bit of information: the first line. In the above example, it tells us that the web browser is requesting the root (**/**) of our web page. The *Sed* script just has to read this and reply with an appropriate HTTP response containing our web page.

An HTTP response containing just the text 'hello world' would look like this:

```
HTTP/1.1 200 OK

hello world
```

We only need to send this once, so this is actually the same as our 'hello world' example from earlier. The *Sed* program for this is:

```
sed 's/.*/HTTP\/1.1 200 OK\n\nhello world\n/;q'
```

You can see here that the forward slash has to be escaped with a backslash, and new lines can be introduced with **\n**. If you put this into **run.sh** in place of the **cat** line, you should find that you can reload **localhost:8889** in your browser and see the 'hello world' message. There's no need to stop and restart **server.sh**: as soon as the changes to **run.sh** are saved, any new requests will go to the new script.

TCP streams are a little different to files, because they don't terminate in the same way. When the server sends the HTTP request, it doesn't close the stream, because if it did, it wouldn't be able to get the response. Instead, it expects the server to close the stream once the entire HTTP response is sent. Because of this, we have to explicitly quit *Sed* before the browser will display the data. If we used the alternative *Sed* hello world (based on ranges), then our server would send the data to the browser, but the browser wouldn't load it because it'd be constantly waiting for more HTTP data from the server.

The only difference between the above response and a web page response is that a web page is written in HTML. We could put all of our web page HTML in the Sed line, but it would rapidly become unmanageable. To make like a little easier, we'll allow ourselves to use *Bash* variables as well as *Sed*. A simple HTML hello world can then be done with:

```
httpheader="HTTP\/1.1 200 OK \n\n"
htmlheader="<html><head><title>hello<\/title><\/head>"
htmlbody="<body><h1>Hello World<\/h1><\/body><\/
html>"
sed "s/.*/$httpheader$htmlheader$htmlbody/;q"
```

We have to escape the forward slashes, but otherwise, we can pass normal HTML straight to *Sed*.

We've now created a web server capable of serving a single page. This is fairly impressive given the limitations of *Sed*, but it's far short of the web framework we set out to create. In order to be able to serve up different pages, we need to be able to run multiple search and replace rules. We can do this in *Sed* by putting each rule on a separate line, but we can't do it with our current setup because we end the rule with the quit (**q**) command. Using this, it will exit *Sed* whether the line is matched or not after the first rule. *Sed* can selectively run blocks of code only on lines that match a particular regular expression using the format:

```
          /<regex>/ { <sed code> }
```

We can now write our code to serve up different pages when the browser requests different URLs in different code blocks. The below code serves up two different pages, one at **localhost:8889/test** and one at **localhost:8889/helloworld**. Since both of these will quit *Sed* if they match the URL, we know that we'll only reach the bottom of the *Sed* script if none of the URLs match. If this is the case, we'll serve up a 404 error message.

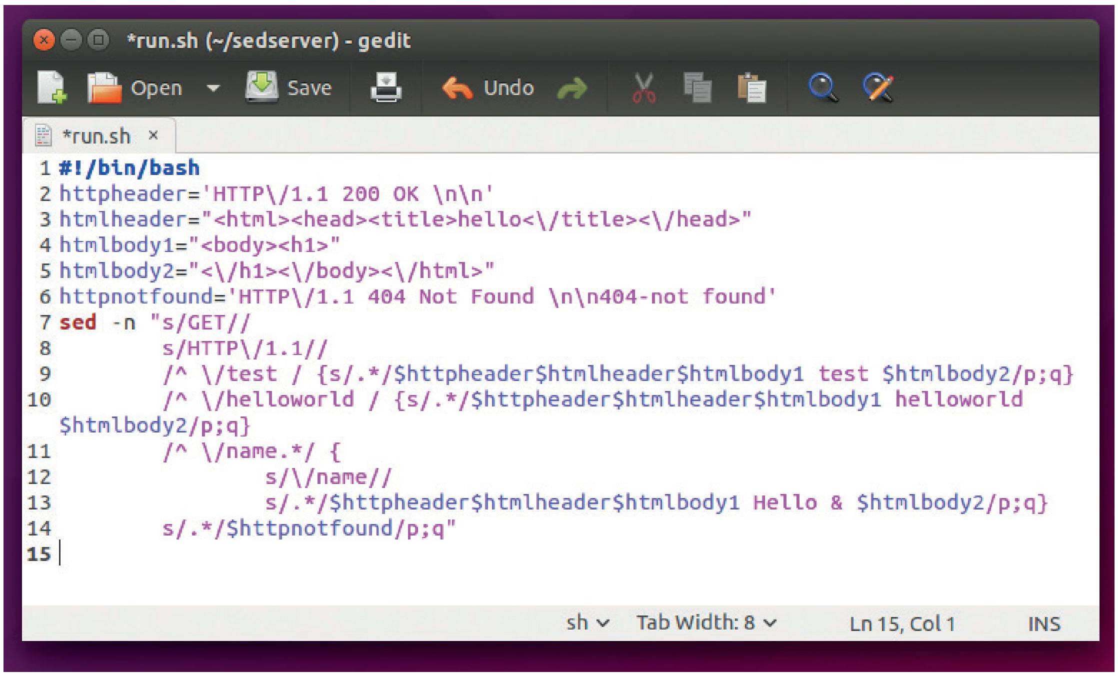
```
httpheader='HTTP\/1.1 200 OK \n\n'
htmlheader="<html><head><title>hello<\/title><\/head>"
htmlbody1="<body><h1>"
htmlbody2="<\/h1><\/body><\/html>"
httpnotfound='HTTP\/1.1 404 Not Found \n\n404-not
found'
```

Extra Sed features

We've focussed on the features of *Sed* that we need to make our web framework. It just so happens that this also covers the parts of *Sed* that make is useful for text processing, but there are a couple of useful features that we've overlooked. These are:

- The **-i** command line flag is used to edit a file in place. With this, the changes are written back to the source of the text rather than send to standard output.
- The **g** flag to the **s** command applies the transformation to all occurrences of the regular expression on a line rather than just the first. For example:

```
$ echo 'eeeee' | sed 's/e/f/'
feeee
$ echo 'eeeee' | sed 's/e/f/g'
fffff
```



```
#!/bin/bash
httpheader='HTTP\/1.1 200 OK \n\n'
htmlheader="<html><head><title>hello<\/title><\/head>"
htmlbody1="<body><h1>"
htmlbody2="<\/h1><\/body><\/html>"
httpnotfound='HTTP\/1.1 404 Not Found \n\n404-not found'
sed -n "s/GET//
        s/HTTP\/1.1//
        /^ \/test / {s/.*/$httpheader$htmlheader$htmlbody1 test $htmlbody2/p;q}
        /^ \/helloworld / {s/.*/$httpheader$htmlheader$htmlbody1 helloworld
$htmlbody2/p;q}
        /^ \/name.*/ {
                s/\/name//
                s/.*/$httpheader$htmlheader$htmlbody1 Hello & $htmlbody2/p;q}
        s/.*/$httpnotfound/p;q"
```

Who would have thought you could build a complete web framework in 14 lines of *Bash* and *Sed* ?

```
sed -n "s/GET//
        s/HTTP\/1.1//
        /^ \/test / {s/.*/$httpheader$htmlheader$htmlb
ody1 test $htmlbody2/p;q}
        /^ \/helloworld / {s/.*/$httpheader$htmlheader
$htmlbody1 helloworld $htmlbody2/p;q}
        s/.*/$httpnotfound/p;q"
```

The first two lines of this code strip out the **GET** and the **HTTP/1.1** from the request by replacing them with empty strings. This isn't completely necessary, but it makes our URL matching a little less cluttered.

The caret character (^) at the start of the regular expression that matches **/test** and **/helloworld** is there to match the start of the line. Matching in this way means that there can't be an accidental match against another URL that also includes the string **/test**. There's also a space after the URL, which means that the URL will only match if it's complete, as spaces aren't allowed in URLs. (Actually, the situation is a little more nuanced than this. While you can't have a space character in a URL, you can encode a space as **%20**, which your browser will display as a space, so even though it sometimes looks like there are spaces in URLs, the space is never sent to the server in a HTTP request).

This *Sed* web server can be expanded to serve up as many pages as you like, and although our pages are quite simple (to save space in the magazine), they could be as complex as you like. The *Bash* variables make it quite easy to reuse different parts of the page (such as, for example, creating a standard sidebar), however, it's still not quite a web framework. We should enable our server to create dynamic content based on values that the user supplies. A simple example of this is where a browser goes to the URL **localhost:8889/nameben** and the server responds with **hello ben**. In order to do this, we have to be able to capture part of the URL and include it in the response. *Sed* allows us to do this using back-references. If you put an ampersand character in the **replace** section of an **s** line, *Sed* will put in the entire text that the regular expression matched. For this to work properly, we first have to strip out everything from the URL that we don't want included in the final page. Our final *Sed* script is:

```
sed -n "s/GET//
        s/HTTP\/1.1//
        /^ \/test / {s/.*/$httpheader$htmlheader$htmlb
ody1 test $htmlbody2/p;q}
        /^ \/helloworld / {s/.*/$httpheader$htmlheader
$htmlbody1 helloworld $htmlbody2/p;q}
        /^ \/name.*/ {
                s/\/name//
                s/.*/$httpheader$htmlheader$htmlb
ody1 Hello & $htmlbody2/p;q}
        s/.*/$httpnotfound/p;q"
```

This simple *Sed* web framework could easily be expanded to serve complex web apps, though it's probably best to stick to a more conventional web framework for this. We created this as a fun way to learn *Sed*, not as a safe, secure web server.

> We created this as a fun way to learn Sed, not as a safe, secure web server

Technically, it's not written completely in *Sed*, since it requires a little help from *Bash* and *Netcat*, but *Sed* does all the heavy lifting. The small codebase shows just how powerful *Sed* can be once you properly understand the paradigm of stream editing. Using the techniques we've used here, you could easily convert a logfile into a report, tidy up your source code, or do all manner of text processing tasks.

Ben Everard is porting linuxvoice.com to Sed. He expects to finish the project shortly after hurd 1.0 is released.

SCRIPTING LANGUAGES: WHAT AND WHY

Why go to the extra effort of compiling your code?

JULIET KEMP

WHY DO THIS?

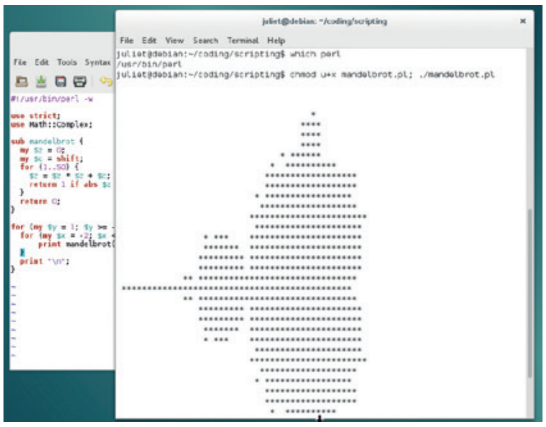
- Pick the right language for you
- Spread information asynchronously over the web

Scripting languages seem to be everywhere these days; but what exactly is a scripting language? Ultimately, it's not so much any feature of the language itself, as what it's used for: creating and running scripts (roughly, an automated series of commands).

This generally means that a language used for scripting will be interpreted rather than compiled, and dynamically typed. (In theory you could write a 'script' in pretty much any language, but Java, for example, would be a bad fit.) The usage-based definition does mean that the line between a scripting language and a general-purpose language is a little vague. A 'script' usually means a single piece of code, which runs start to finish, but general-purpose high level scripting languages like Perl, Python, and Ruby can be used to write 'scripts' that are thousands of lines long and have many different components.

As a rule, scripting languages are designed to be fast to learn and fast to write. Being interpreted languages (in general; Ruby can be either, depending on its implementation), however, they're likely to be slower to execute than a compiled language. Their increasing popularity reflects gradual improvements in computing hardware. The faster a computer runs, the less the speed difference between interpreted and compiled programs matters; and the more important the development speed becomes. Missing out the compiling step makes for a much faster development cycle, which also fits well with modern programming practices like XP. The larger and more complex the software, though, the more sensible it may become to use a compiled language again.

> The increasing popularity of scripting languages reflects gradual improvements in computer hardware



juliet@debian: ~/coding/scripting
File Edit View Search Terminal Help
File Edit Tools Syntax

```
#!/usr/bin/perl -w

use strict;
use Math::Complex;

sub mandelbrot {
  my $z = 0;
  my $c = shift;
  for (1..50) {
    $z = $z * $z + $c;
    return 1 if abs $z
  }
  return 0;
}

for (my $y = 1; $y >= -
  for (my $x = -2; $x <
    print mandelbrot(
  }
  print "\n";
}
```

```
juliet@debian:~/coding/scripting$ which perl
/usr/bin/perl
juliet@debian:~/coding/scripting$ chmod u+x mandelbrot.pl; ./mandelbrot.pl
```

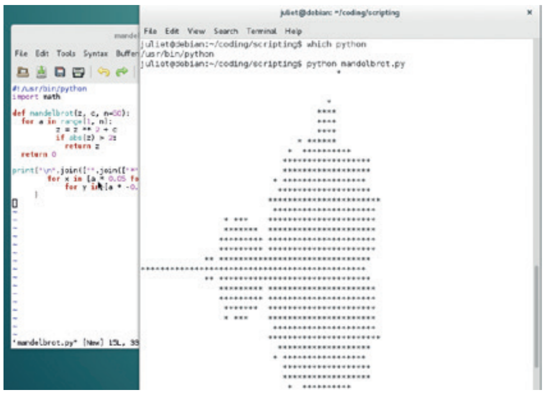
Running the Perl script

The big-hitters in the scripting language world are high-level general-purpose dynamic languages, often thought of as 'glue' languages, connecting things together. Many other scripting languages are domain-specific, such as Sed or Awk, or (rather more up-to-date) JavaScript. JavaScript is an embedded language, used only within a particular application or set of applications. Emacs Lisp is another application-specific language, and many games have their own scripting language or dialect. Finally, job control languages and shells, like *Bash*, are another form of scripting language, which are often also used as glue languages and command-line interpreters.

The Start of Scripting

Early mainframes didn't have significant (or, sometimes, any) direct user access; instead, they batch processed jobs. To make this easier, various languages including IBM's Job Control Language were developed.

Batch processing languages were followed in the 1960s by interactive shells, and shell scripts to automate running programs. These accreted more



juliet@debian: ~/coding/scripting
File Edit View Search Terminal Help
File Edit Tools Syntax Buffers
"mandelbrot.py" [New] 15L, 35

```
#!/usr/bin/python
import math

def mandelbrot(z, c, n=50):
  for a in range(1, n):
        z = z ** 2 + c
        if abs(z) > 2:
            return z
  return 0

print("\n".join(["".join(["*
        for x in [a * 0.05 fo
            for y in [a * -0.
    ]
```

```
juliet@debian:~/coding/scripting$ which python
/usr/bin/python
juliet@debian:~/coding/scripting$ python mandelbrot.py
```

Running the Python script

and better features as programmers hacked on them. TRAC, by Calvin Mooers, invented the idea of command substitution, which interprets a command within a script and uses its output in the containing script – like the backtick operator in a modern shell.

Stuart Madnick at MIT wrote the CMS EXEC scripting language (originally called COMMAND) for the IBM VM/CMS OS in the late 60s. This included control statements of various types (including conditional statements and loops) and a few built-in functions. You can see the bones of modern scripting languages beginning to emerge.

Perl

Larry Wall released version 1.0 of Perl in 1987. His aim was to create a general-purpose Unix scripting language to make report processing easier. Improvements in computer hardware meant that efficient programming practices were becoming more important than super-efficient code, and Wall wanted to support this. Perl's ancestors include Awk (1977), and the Unix shell sh (1978), and it predates Linux (1991) by several years. (See **http://history.perl.org/PerlTimeline.html** for a cool timeline including Unix/Linux and newsgroup history.)

Perl 2 (1988) and Perl 3 (1989) followed quickly. Perl 4 (1991) got its version number bumped solely to clearly identify it as the version documented by *Programming Perl* (aka the Camel Book), the canonical Perl reference.

Perl 5 (1994) included a comprehensive rewrite of the interpreter, and a whole bunch of new features, including objects and module support. This in turn gave rise to CPAN, the Comprehensive Perl Archive Network, created as a Perl and Perl module repository in 1995. CPAN is both an incredible resource, and a place to get catastrophically lost; it has incredible 157,742 (at time of writing) modules, but the standard of those modules varies dramatically, and it can be hard to navigate and to find high-standard modules. (The MetaCPAN project and the Task::Kensho module from the Enlightened Perl Organisation are attempts to make this easier.)

Perl 6 is in active development, but is now considered to be a separate language; backwards compatibility with Perl is not a goal, but Perl 6 is supposed to be recognisably "a Perl programming language". The changes are largely aimed at normalising the language. It's been in development since 2000 and still isn't fully ready.

Perl is still in active use, and will doubtless remain so for a long time. However, it's less popular than it used to be, especially for new projects. The Perl motto, There's More Than One Way To Do It, still sums up the immense flexibility which is both Perl's biggest advantage and its biggest disadvantage.

PRO TIP

sed and awk might be domain-specific languages, but if you want to see some general-purpose sed and awk code, check out the Rosetta Code website!

Perl code

Nearly all Linux systems will already have Perl installed; if not, get it via your package manager. Here's Hello World:

```
#!/usr/bin/perl -w
print "Hello World\n";
```

#! is the shebang, which tells the system to treat the rest of the line as the interpreter and pass the script pathname into it. So when we run **hello.pl**, the first line tells the system to fire up **/usr/bin/perl**, and feed **hello.pl** in to be interpreted. Language interpreters ignore the shebang, either because **#** is a comment indicator, or as a special case.

The **-w** switch turns on warnings, which is good practice. (You can also add the line **use strict;** just underneath that, to catch certain types of compile- and run-time errors in advance.) You'll see that Perl statements end with **;**. Make the file executable and run it with **./hello.pl**.

This slightly more interesting code outputs the Mandelbrot set to your terminal screen:

```
#!/usr/bin/perl -w
use strict;
```

The Mandelbrot Set

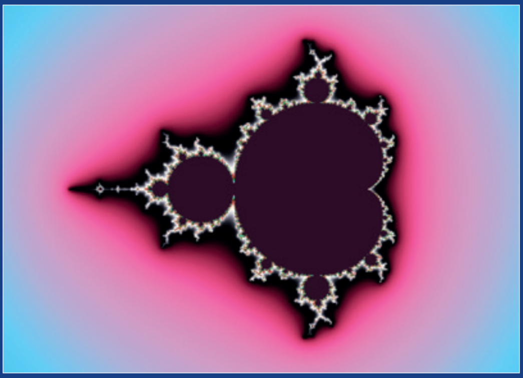
The Mandelbrot Set is defined on the plane of complex numbers (represented as an (x,y) grid). For each complex number **c**, iterate:

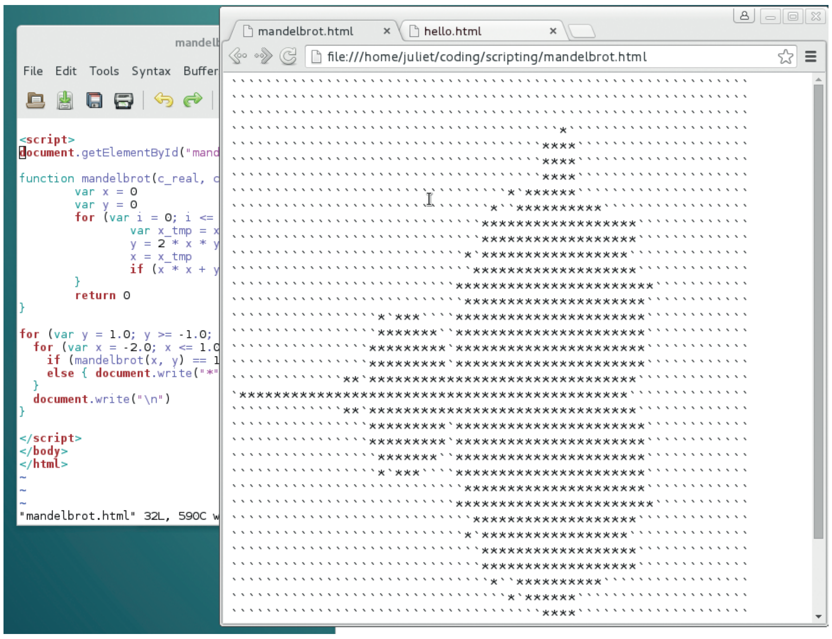
```
zk+1 = zk2 + c
```

If the iterative sequence stays close to **c**, then **c** is in the Mandelbrot set. If the sequence spirals off and away, **c** is not in the set.

When calculating this, we'll iterate the sequence 50 times, and if **zk+1** leaves a circle of radius 2 around **c**, we'll conclude that **c** is not in the set.

Our grid will have the x axis running from -2 to 1, and the y axis running from -1 to 1. We'll draw it as ASCII art, starting from the top-left corner (x=-2, y=1), one y-line at a time. Each ASCII character (space or *) will represent a move of 0.05 along the grid. You can change the hard-coded values to see what happens, or add colour to make it more visually complex.


With an appropriate program, you can generate the set and then zoom in to see more and more fractal detail. Licence: CC-SA


mandelbrot.html
hello.html
file:///home/juliet/coding/scripting/mandelbrot.html
File Edit Tools Syntax Buffer

```
<script>
document.getElementById("mand
function mandelbrot(c_real, c
        var x = 0
        var y = 0
        for (var i = 0; i <=
                var x_tmp = x
                y = 2 * x * y
                x = x_tmp
                if (x * x + y
        }
        return 0
}

for (var y = 1.0; y >= -1.0;
  for (var x = -2.0; x <= 1.0
    if (mandelbrot(x, y) == 1
    else { document.write("*"
  }
  document.write("\n")
}

</script>
</body>
</html>
~
~
~
"mandelbrot.html" 32L, 590C w
```

Bugfixing can be difficult as errors don't show up on the displayed page; try JSLint (**www.jslint.com**) or the JavaScript console in your browser to help.

```
use Math::Complex;  # deals with complex numbers
sub mandelbrot {
  my $z = 0;
  my $c = shift;
  for (1 .. 50) {
    $z = $z * $z + $c;
    return 1 if abs $z > 2; # if it's bigger than two it
escapes
  }
  return 0;
}
for (my $y = 1; $y >= -1; $y -= 0.05) {
  for (my $x = -2; $x <= 1; $x += 0.05) {
    print mandelbrot($x + $y * i) ? ' ' : '*';
  }
  print "\n";
}
```

Functions/subroutines are identified with sub name. Parameters are passed in as a list, and can be retrieved by Shift-ing off the list one at a time. However, they don't have to be identified in the function name. The subroutine here returns non-zero if the number is not part of the Mandelbrot set, and zero if it is.

for loops have a three-part structure: **for** (**start**, **end**, **increment**) {..}. You can increment or decrement in a **for** loop, and you can nest them, as here.

PRO TIP

Perl's unofficial backronym is Practical Extraction and Reporting Language, though Wall prefers his own Pathologically Eclectic Rubbish Lister.

The **?:** operator shown here is a shorthand way of writing if-then-else. If the function call here returns **true** (the value escapes, so is not in the set), we print an empty space; otherwise (**else**), we print an asterisk.

Python

Python is nearly as old as Perl; Guido van Rossum (Python's Benevolent Dictator for Life) started implementing it in 1989, and released version 0.9.0 in 1991. It had classes and class inheritance, functions, core datatypes, and a module system. In 1994, version 1.0 was finally released, including the **lambda** and **map**, **filter**, and **reduce** functions, courtesy of a Lisp hacker's patches. Initially, Python's clean syntax was intended to make it accessible to non-programmers; these days it is still designed to be accessible and easy to learn, but non-programmers are no longer a specific target market.

Python is highly extensible, and modules can be written in either Python or C; which meant that Van Rossum's C programmer colleagues could start working with it straight away. One of Van Rossum's aims was to create a bridge between the shell and C; a second language for C/C++ programmers to use in situations where C would be overkill. Python's ancestors include ABC (a language intended for non-programmers), C, Bash, Lisp, Perl, and Java.

Version 2.0, in 2000, introduced list comprehensions, an idea borrowed from functional programming that enables you to easily create lists. Version 3.0, released in 2008, was a bigger change, aiming to reduce the redundancy that had accumulated in previous versions of Python (and ditching backwards-compatibility in the process). In contrast to the Perl approach, the design philosophy behind Python 3 was "there should be one – and preferably only one – obvious way to do it". But it retained its multiple paradigms, an approach common to general-purpose scripting languages; you can still write object-oriented Python, structured Python, functional Python, and so on. This flexibility is great, but it can make it hard, especially for beginners, to get to grips with other people's code.

Python code is compact (you need fewer lines of code to do the same amount of work than, say, Java) and readable, runs fast, is quick to develop (like other scripting languages), works well with object-oriented programming, and has broad applicability. However, it's still slower than compiled languages, though optimisation can help, and it's not a client-side web language. The huge recent increase in the number of libraries available has greatly helped its uptake; it's always been accessible for beginners but now it is more powerful for pros.

There's a fascinating interview with Van Rossum at **www.artima.com/intv/pythonP.html** on the beginnings of Python.

Python code

You'll probably have Python already on your Linux system too (if not, check your package manager). Here's Hello World:

```
#!/usr/bin/python
print "Hello World"
```

You don't need to specify the newline at the end of the output string. Nor do you need a semicolon to terminate the statement, as Python uses whitespace to terminate statements.

Here's the Mandelbrot Set code in Python:

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

Bash

Bash is slightly different to Perl, Python, and JavaScript, being a job control/command execution languages. But it's still a complete language, although not a great choice for more than a page or two of code.

The Bourne shell, *sh*, was released in 1977 in Version 7 of UNIX, for use as an interactive command interpreter and scripting language. It became the default UNIX shell and was used for a whole host of practical jobs across the system.

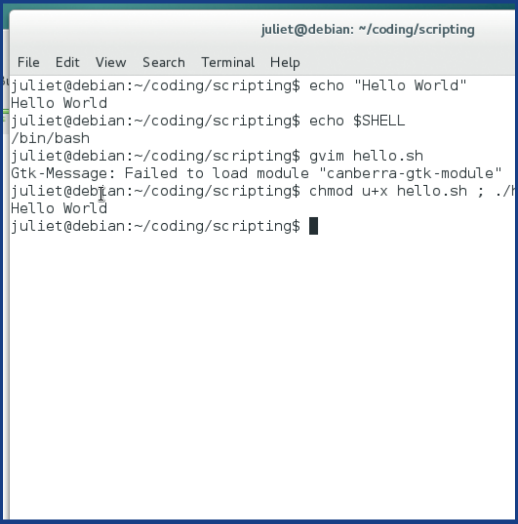
The GNU project, aiming to produce an entirely free software system, needed a FOSS drop-in replacement for *sh*. Brian Fox, funded by the Free Software Foundation, released *Bash* (standing for Bourne-again shell) in beta in 1989. It's been the standard Linux shell ever since (though shells such as *tcsh* and *zsh* are also popular). To check which shell you're using, type **echo $SHELL**. If it's something else, type **/bin/bash** to try *Bash*.

Bash handles command-line input:

```
$ echo "Hello World"
```

Alternatively, you can create a file:

```
#!/bin/bash
echo "Hello World"
```


juliet@debian: ~/coding/scripting
File Edit View Search Terminal Help

```
juliet@debian:~/coding/scripting$ echo "Hello World"
Hello World
juliet@debian:~/coding/scripting$ echo $SHELL
/bin/bash
juliet@debian:~/coding/scripting$ gvim hello.sh
Gtk-Message: Failed to load module "canberra-gtk-module"
juliet@debian:~/coding/scripting$ chmod u+x hello.sh ; ./h
Hello World
juliet@debian:~/coding/scripting$ 
```

Bash doesn't require semi-colons, but instead treats a newline as starting a new command.

```
import math
def mandelbrot(z, c, n=50):
  for a in range(1, n):
    z = z ** 2 + c
    if abs(z) > 2:
      return z
  return 0
print("\n".join(["".join(["*" if not mandelbrot(0, x + y * 1j)
else " "
  for x in [a *  0.05 for a in range (-40, 20)]])
   for y in [a * -0.05 for a in range (-20, 20)]])
  )
```

As with Perl, we import a library (here the **math** library) to handle complex numbers. In Python you write x + yi as x + y * 1j.

Functions are defined with the **def** keyword. **for** loops use the **range** keyword. The value in the range must be an integer, but if, as here, you want a non-integer loop, you can multiply the **range** value by a non-integer and feed that into your loop value.

As Python automatically adds a newline after a print statement, to get the output we want we have to use **join**. Note that the **for** loops are inside the **print**/**join** statement, and also the use of **if not … else**. This uses list comprehensions to create a list of all the outputs from the Mandelbrot function, and join them together.

JavaScript

JavaScript was, famously, developed in 10 days in 1995 by Brendan Eich at Netscape. Netscape at the time was offering a server with a portable version of Java to run server-side applets, and wanted an easier, interpreted, language to go alongside it, aimed at non-programmers. Initially it was called LiveScript and first shipped in *Netscape Navigator* in 1995, but it was later renamed JavaScript, at the same time as Netscape added Java support to *Navigator*, possibly as a marketing ploy. It's interpreted, client-side, and is executed directly in the web browser.

Microsoft reverse-engineered JavaScript to produce JScript, which they released in 1996 as part of IE3 and IIS. The differences between the two implementations made it hard to design websites that worked well in both browsers, and JavaScript began to get a bad reputation for blocking cross-browser support. Netscape submitted it to the international standards organisation ECMA, and the ECMAScript official standard was released in 1997. JavaScript is the best known implementation of this standard; ActionScript 3 is another.

JavaScript relies on the browser providing objects and methods to enable it to interact with the browser environment. This does make for potential security risks, which is part of why JavaScript has had problems in the past. Bad coding can make users vulnerable to malicious scripts, and there have been a lot of problem developers and companies.

This was perhaps made worse because JavaScript was initially seen as the province of 'amateur' creators of websites, not professional programmers. Many of these 'amateurs' just pasted someone else's scripts straight into their website, without checking the quality or security of those scripts.

However, with the rise and rise of the web, and helped by the introduction of Ajax (Asynchronous JavaScript and XML: basically a way to communicate with the server asynchronously, thus updating a page without having to refresh it, which is great from a user experience perspective), JavaScript has become a more respected, and more professional, language. Dynamic web apps and web pages rely heavily on JavaScript -- it's nearly essential to use JavaScript if you want your webpage to look remotely up-to-date. Node.js has also finally allowed JavaScript to jump from client-side to server-side, and the new HTML 5 APIs offer more options for controlling webpages, making it ever more useful – as long as you're careful with the security.

JavaScript code

You can't run JavaScript from the command line; to try it out, open this **hello.html** page in your browser.

```
<!DOCTYPE HTML>
<html>
<head></head>
<body>

<script>
alert("Hello World")
document.write("Hello World")
</script>
</body>
</html>
```

This very minimal HTML just runs the script, wrapped in a **<script>** tag. In fact, this says Hello World twice: the **alert** line pops up an alert dialog (with an OK button automatically attached); the **document.write** line writes it as HTML body text. Like Python, JavaScript doesn't generally require (but will accept) a semicolon to end a statement, but there is some debate about whether it's better to include them or not. You can also write a separate **file.js** script and include it in the HTML.

Stats from 2015 suggest that JavaScript, Python, Ruby and PHP are all in the top 10 most popular languages

Here's the Mandelbrot set:

```
// HTML and head tags to start file as above
<body id="mandelbrot">
<script>

document.getElementById("mandelbrot").style.
fontFamily = "courier";

function mandelbrot(c_real, c_img) {
  var x = 0
  var y = 0
  for (var i = 0; i < 50; i++) {
    var x_tmp = x * x - y * y + c_real
    y = 2 * x * y + c_img
    x = x_tmp
    if (x * x + y * y >= 4) { return 1 }
  }
  return 0
}

for (var y = 1.0; y >= -1.0; y -= 0.05) {
  for (var x = -2.0; x <= 1.0; x += 0.05) {
    if (mandelbrot(x, y) == 1) { document.write("`") }
    else { document.write("*") }
  }
  document.write("\n")
}

</script>
</body>
</html>
```

As in the first line of the script, you can set document values in JS. (Note that the body element has to have an **id** value for this line to work.)

Functions in JS use the **function** keyword, and you can pass in multiple parameters.

As with Python, JS variables don't need any signifiers, but should be declared with **var** the first time they're used (though this isn't essential).

for loops work in very much the same way as the other two languages. Since JS lacks a complex number library, we have to fake it by handling the real and complex parts separately. **document.write** didn't want to output whitespace, so the space is a backtick here instead.

If you'd prefer to make better use of HTML and draw a prettier (non-ASCII) version of this, there's some code on the Rosetta Code website (**rosettacode.org**). And, as with Perl and Python, there are plenty of online resources if you want to explore JavaScript further.

And there's more...

The advantage of scripting languages is their speed of development, and a big part of that is their dynamic typing. The popularity of scripting languages is thus arguably associated with the rise of unit tests in modern programming; unit tests make type safety less valuable, because they (should...) pick up that sort of error, and others besides. Dynamic languages are faster to write, the lack of compile time with interpreted languages makes it faster still, and the speed of modern computer hardware means that run speed (which is slower in interpreted languages) is less important. Or that's the theory; it certainly doesn't apply to every project.

GitHub and StackOverflow stats from early 2015 suggest that JavaScript, Python, Ruby, and PHP are all in the top 10 most popular languages, more than holding their own with Java and C/C++. Job surveys also show much the same thing. (Stats compiled by **www.sitepoint.com/whats-best-programming-language-learn-2015** and **www.codingdojo.com/blog/8-most-in-demand-programming-languages-of-2015**.) Web 2.0 and associated new technologies place an emphasis on scripting langauges, and new languages such as Lua, a very light and adaptable, dynamically typed, general embeddable extension language, are on the rise. Scripting languages may not, contrary to some of the hype, be about to take over the programming world, but 30 or more years in, they're more important and more flexible than ever.

Juliet Kemp is a computing polyglot, having fun with coding one language at a time.

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

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

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.

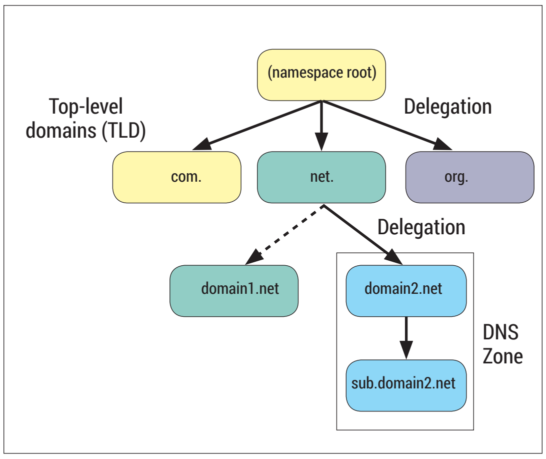
Domain names

Explore the nuts and bolts of DNS, the quiet workhorse of the internet that too many of us take for granted.

While many people think of DNS exclusively in the context of mapping host names to IP addresses, in fact, "DNS is a general (if somewhat limited) hierarchical database, and can store almost any kind of data, for almost any purpose". By analogy, responding to DNS queries often involves talking to several sources. In this Core Tech, we'll learn how it happens, both internet-wide and in your home network.

Once upon a time, there was a single file named **HOSTS.TXT**. It contained names for all hosts on the internet, and the network information centre (NIC) kept it current. Connected hosts updated this file via FTP. Everything went fine, until things started to grow big. Needless to say, the **HOSTS.TXT** approach didn't scale well. Networks of the 80s had limited capacity, and **HOSTS.TXT** updates consumed their scarce bandwidth. A new system (DNS) had to be global, distributed to handle increasing load, and generic, to support emerging applications. Today, DNS facilitates name resolution, but also spam protection and service discovery, among other things.

DNS builds on the concept of a hierarchical, single-rooted namespace. A section of this tree (spanning one or more domains) is called a "zone".



DNS namespace in a nutshell. Different colours represent separate zones, solid arrows indicate domain delegation.

A name server can handle one or more zones, and a single zone is often serviced by multiple servers for reliability reasons. A server that stores data for a given zone is referred to as being "authoritative" for it. Other name servers may cache its responses and re-use them in their own answers; those answers are non-authoritative. They are no evil; in fact, we often use name servers on wireless routers that aren't authoritative for anything. Sometimes, these servers are called "caching".

A domain name consists of one or more dot-separated labels. Names read right to left: **com**, **linuxvoice**, **www**. In fact, each name ends with the dot that separates an empty label denoting namespace root. User-friendly resolvers (that is, programs that resolve names) don't force us to type it, but many other tools prefer this final dot explicit. A domain name that starts at root is called "fully qualified" (FQDN); everything else is unqualified or relative. When Linux encounters an unqualified name, it tries to make it into an FQDN. To do so, it appends domains from the system-configured search list. So, **www** on a corporate network may open company's website, as **company.com** often comes first in the search list. Search lists are convenient, but may cause resolution slowdowns if configured improperly.

On a good record

As the DNS is the database, it can also be viewed as a set of records. A resource record (RR) essentially maps a name to some data (RDATA). Each RR also has a class (typically 'IN', or Internet), a TTL (time to live, a number of seconds for the record to stay valid in cache), and, last but not least, type. There are many RR types defined, and in this Core Tech we'll cover most important ones.

Let's begin with 'A'. Records of this type map domain names to IPv4 addresses. A's cousin, 'AAAA', does the same for IPv6:

```
www.linuxvoice.com.    300    IN    A    104.28.6.18
www.linuxvoice.com.    300    IN    A     104.28.7.18
```

The above is standard textual representation of an RR. It begins with the owner's name (note the trailing dot), followed by TTL, class, type and RDATA, which is just an IP address in this case. Note that **www.linuxvoice.com** really has two IPv4 addresses. This yields a simple load balancing scheme: one connecting client will resolve it as 104.28.6.18, while another will get 104.28.7.18. It would spread the load on two boxes. Also note that addresses may change by the time you read this, yet names will stay the same. DNS provides an abstraction that makes IP address changes transparent to end users.

The MX (Mail eXchange) record is what mail agents use to find a host accepting email for a given domain:

```
linuxvoice.com.     300    IN    MX     10 smtp.
linuxvoice.com.
```

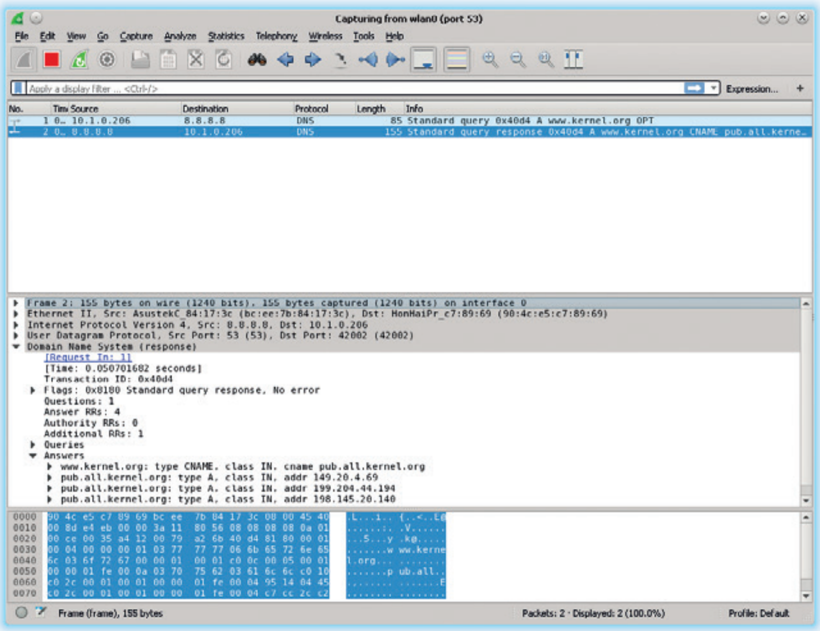
Here, RDATA is the host name and 16-bit number (10) dubbed priority. A mailer should contact a highest-priority (lowest numbered) mailserver first. So, if one wants to deliver mail for **info@linuxvoice.com**, it should talk to **smtp.linuxvoice.com**, port 25/tcp.

Now, consider what **www.kernel.org** really is:

```
www.kernel.org.     599   IN    CNAME   pub.all.kernel.org.
pub.all.kernel.org.   599   IN    A          198.145.20.140
```

CNAME stands for Canonical Name. Put simply, this record says that **www.kernel.org** is just an alias for **pub.all.kernel.org**, which resolves to 198.145.20.140. Note that IP address comes from a separate 'A' record.

DNS also supports reverse mapping, or finding a name by IP address. Naturally, there are hosts that don't have a registered name, but those that do should also have a **PTR** record in the **in-addr.arpa** domain:



```
Capturing from wlan0 (port 53)

No.   Tim Source        Destination   Protocol  Length  Info
  1 0… 10.1.0.206       8.8.8.8       DNS           85  Standard query 0x40d4 A www.kernel.org OPT
  2 0… 8.8.8.8          10.1.0.206    DNS          155  Standard query response 0x40d4 A www.kernel.org CNAME pub.all.kerne…

▶ Frame 2: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface 0
▶ Ethernet II, Src: AsustekC_84:17:3c (bc:ee:7b:84:17:3c), Dst: HonHaiPr_c7:89:69 (90:4c:e5:c7:89:69)
▶ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 10.1.0.206
▶ User Datagram Protocol, Src Port: 53 (53), Dst Port: 42002 (42002)
▼ Domain Name System (response)
    [Request In: 1]
    [Time: 0.050701682 seconds]
    Transaction ID: 0x40d4
  ▶ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 0
    Additional RRs: 1
  ▶ Queries
  ▼ Answers
    ▶ www.kernel.org: type CNAME, class IN, cname pub.all.kernel.org
    ▶ pub.all.kernel.org: type A, class IN, addr 149.20.4.69
    ▶ pub.all.kernel.org: type A, class IN, addr 199.204.44.194
    ▶ pub.all.kernel.org: type A, class IN, addr 198.145.20.140

0000  90 4c e5 c7 89 69 bc ee  7b 84 17 3c 08 00 45 40   .L...i.. {..<..E@
0010  00 8d e4 eb 00 00 3a 11  80 56 08 08 08 08 0a 01   ......:. .V......
0020  00 ce 00 35 a4 12 00 79  a2 6b 40 d4 81 80 00 01   ...5...y .k@.....
0030  00 04 00 00 00 01 03 77  77 77 06 6b 65 72 6e 65   .......w ww.kerne
0040  6c 03 6f 72 67 00 00 01  00 01 c0 0c 00 05 00 01   l.org... ........
0050  00 00 01 fe 00 0a 03 70  75 62 03 61 6c 6c c0 10   .......p ub.all..
0060  c0 2c 00 01 00 01 00 00  01 fe 00 04 95 14 04 45   ........ .......E
0070  c0 2c 00 01 00 01 00 00  01 fe 00 04 c7 cc 2c c2   ........ ........

Frame (frame), 155 bytes          Packets: 2 · Displayed: 2 (100.0%)          Profile: Default
```

Wireshark network analyser can easily decode both DNS queries and their responses.

```
140.20.145.198.in-addr.arpa. 5  IN     PTR     tiz-korg-pub.
kernel.org.
```

PTR is for Pointer; such records just point to some other RR. To do a reverse mapping, reverse the IPv4 address octets and append the **in-addr.arpa.** suffix. Recall that right-to-left is the natural DNS label processing order. So, one can map IPV4 subnets to DNS zones and delegate them to different servers.

Resource records may even contain some arbitrary data. The TXT type is reserved for that. Several technologies rely on TXT records, including DKIM and SPF, which help fight spam.

I hope you'd agree now that DNS is little bit more than an internet-scale phonebook. If so, let's learn how DNS questions get their answers.

A day with DNS resolver

DNS messages are usually UDP datagrams coming from (or sent to) port 53. A maximum message size is 512 bytes. TCP connections are supported as well. However, they are mainly useful for data-intensive operations, like zone transfers which happen between name servers.

RFC 1035 defines a single message format for queries and responses. It includes a header with several bit fields (flags) and four sections: **Question**, **Answer**, **Authority** and **Additional**. **Question** consists of a class, type and name of the record we're looking for. The **answer** section contains records (if any) that directly answers the query. **Authority** refers to authoritative name servers for the target domain. It doesn't mean that any of this servers answered the query; if so, an **AA** bit would be set in the response flags. An additional section provides related records that the name server thought would be helpful for the client. For instance, it may contain **A** records for name servers referenced in the **Authority** section.

Queries are resolved recursively. If a server is not authoritative for the domain in query, and it doesn't

Resolving in Glibc

Glibc, the most popular C library in Linux so far, provides its own stub DNS resolver. Available via **gethostbyname()** and related library functions, it comes as a part of the Name Service Switch (NSS) subsystem.

NSS exists because DNS is not the only way to resolve host names in *Glibc*. Other means, say, looking up the **/etc/hosts** file, are also possible. New methods are installed as NSS plugins and configured via **/etc/nsswitch.conf**:

```
hosts: files mdns4_minimal [NOTFOUND=return] dns
```

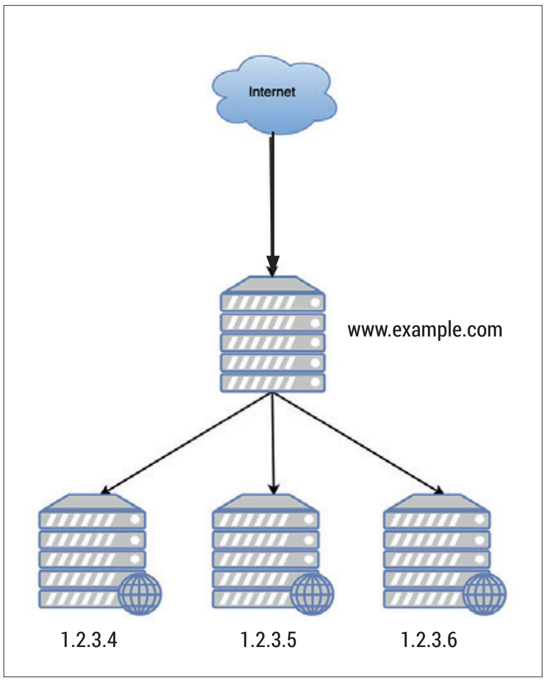
This is the typical configuration for host name resolver. The routine starts at **/etc/hosts** then proceeds to mDNS. If the name is in the **local.** domain, but **nss-mdns** was unable to find it, the procedure terminates. Otherwise, a "normal" DNS query is made. *Glibc* uses another configuration file, **/etc/resolv.conf**, to learn which DNS servers to contact and which domains to try for unqualified names:

```
search company.com
nameserver 10.1.0.1
```

You are unlikely to edit **/etc/resolv.conf** on your Linux desktop, as various tools like *NetworkManager* take care of it automatically. To resolve a name via NSS, use the **getent** command we introduced back in LV023:

```
$ getent hosts www.linuxvoice.com
2400:cb00:2048:1::681c:612 www.linuxvoice.com
```

You see that Linux prefers IPv6 addresses where they are available. **/etc/nsswitch.conf** is also "in charge" for other system names, like users and groups. See **nsswitch.conf(5)** for details.



A simple DNS-based load balancer: connecting clients resolve one hostname to different IPs.

have a relevant record in its cache, it refers to another server that may have the information. It could be an authoritative server for some "intermediate" domain, like **co.uk**, or one of the root DNS servers. A recursive DNS server would follow the referral itself (and cache the response), while non-recursive may simply return it to the client. Server-side recursion is optional, and clients indicate their intention to use it with the **RD** (Recursion Desired) bit in the header. An **RA** bit set in response means Recursion is Available. Iterative clients can do recursion themselves; stub resolvers rely on servers.

Let's do a live experiment. We'll take **dig** and trace how it resolves a name. **dig** is a sophisticated recursive resolver that builds on the same codebase as BIND, "the most widely used name server software".

Resolving a name with **dig** is as simple as typing:

```
$ dig www.linuxvoice.com. +trace
```

The **+trace** bit commands **dig** to trace the domain delegation path from the root. It produces a lengthy output, so I've cut many lines from the samples below.

```
; <<>> DiG 9.10.3 <<>> www.linuxvoice.com. +trace
;; global options: +cmd
.                    142    IN    NS    d.root-servers.net.
;; Received 913 bytes from 192.168.101.1#53(192.168.101.1)
in 21 ms
```

These are **NS** records for root DNS servers. They are authoritative for the '**.**' domain, and play a crucial role in internet operation. Thirteen root DNS servers exist, so their names fit in one DNS message. A cluster of boxes is really running under each name, so real redundancy is even greater.

dig chooses one of these servers, which happens to be from University of Maryland, and asks it for a referral for the **com.** domain:

```
com.              172800  IN    NS    i.gtld-servers.net.
;; Received 742 bytes from 199.7.91.13#53(d.root-servers.
net) in 500 ms
```

VeriSign hosts **com.**, but many of its subdomains are delegated. So **dig** learns that there are two name servers at CloudFlare that are authoritative for the **linuxvoice.com.** subdomain:

```
linuxvoice.com.       172800  IN    NS    heather.ns.
cloudflare.com.
linuxvoice.com.       172800  IN    NS    yichun.ns.
cloudflare.com.
;; Received 677 bytes from 192.43.172.30#53(i.gtld-
servers.net) in 97 ms
```

The only thing left is to ask one of them for the **www.linuxvoice.com.** IPv4 address:

```
www.linuxvoice.com.   300   IN    A    104.28.7.18
www.linuxvoice.com.   300   IN    A    104.28.6.18
;; Received 79 bytes from 173.245.59.248#53(yichun.ns.
cloudflare.com) in 86 ms
```

Phew! That was quite a job, and it's probably why most operating systems (Linux/*glibc* included) today come with a stub resolver.

DNS servers are very important bits of the internet. But in fact, you can create a working DNS system without any servers at all. Moreover, this thing is essential for zero-configuration networking. And it involves lemurs. Prepare to say:

"Bonjour, Avahi!"

Apple calls it *Bonjour*, Linux implementation is codenamed *Avahi*. Either is a piece of software that wraps Multicast DNS (mDNS) and DNS-Based Service Discovery (DNS-SD) protocols.

IP multicasting is like broadcasting in that a single packet reaches multiple recipients. But unlike broadcasting, a host must explicitly subscribe to a multicast group before it gets any data. It is akin to a radio exchange where all peers must tune to the same frequency channel to hear each other. IPv4 reserves addresses in the range 224.0.0.0–239.255.255.255 for multicast traffic.

mDNS uses IP multicast as a primary transport. All mDNS-capable hosts join the 224.0.0.251 multicast group and send messages to port 5353/udp. Peers can see each others' questions and answers and learn from them, effectively working as a distributed DNS server with no central authority.

Name servers galore

Linux isn't short of name server implementations. *BIND* (**https://www.isc.org/downloads/bind**) is the *de-facto* standard. It has many features, and you may think it's a way too much for your small home or office network.

If this is the case, try *Dnsmasq* (**www.thekelleys.org.uk/dnsmasq/doc.html**). This daemon provides all-in-one infrastructure for small networks, including caching DNS and DHCP server. Chances are you already use *Dnsmasq* without even noticing it, as it comes bundled in many wireless routers. *Virt-manager* also relies on *Dnsmasq* to provide name services in its virtual networks. There is also *Unbound* (**www.unbound.net**), a caching DNS resolver library. It provides C and Python APIs, along with a caching recursive resolver daemon built on top of these APIs.

mDNS operates in a special domain, **local.**. It provides flat namespace, so **laptop.local.** is permitted, but **mediaserver.livingroom.local.** isn't, although DNS-SD introduces more levels. mDNS names don't have to be unique. In fact, DNS-SD PTR records are purposely shared. Shared host names don't make any sense though, and there are mechanisms to ensure this doesn't happen.

When an mDNS host becomes online, it sends probe queries for names it wants to be unique. Any host on the net that already owns such a name should promptly reply to "defend" it. If this doesn't happen within a second, the first host assumes there is no conflict and makes an announcement. Should a conflict arise at some later point, there is an arbitration algorithm to solve it.

mDNS also introduces new "ongoing query" mechanics, which fits well into the network browsing use case. The query doesn't stop with the first answer, but repeats at predefined intervals to learn about changes in the network. To advertise support for this feature, mDNS peers send datagrams from port 5353/udp. Otherwise, mDNS messages are almost the same as DNS messages, so you can use **dig** to make mDNS "legacy" (ie non-continuous) queries. Just use 224.0.0.251 as the 'server' and 5353 as target port.

In Linux, two components implement mDNS. **nss-mdns** is an mDNS resolver wrapped as a *Glibc* NSS plugin. Add it to **/etc/nsswitch.conf**, and you'll be able to use **local.** names like any other:

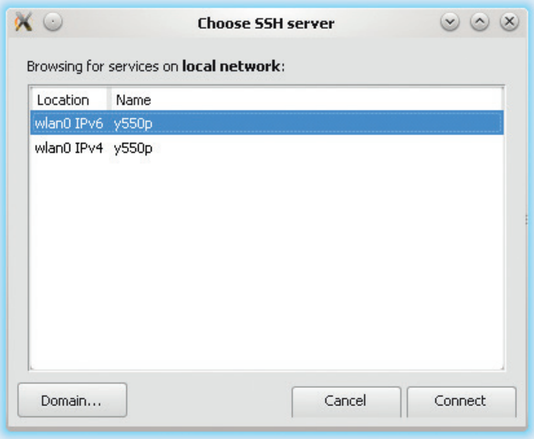
```
[val@y550p ~]$ ping -c 1 y550p.local
PING y550p.local. (10.1.0.206) 56(84) bytes of data.
64 bytes from y550p.local (10.1.0.206): icmp_seq=1 ttl=64
time=0.082 ms
```

Avahi provides an mDNS responder that registers your names and services on the network and answers mDNS queries as they come. Usually, it requires no manual setup: that's the point of zero-configuration, after all. *Avahi* also provides some tools, like **avahi-browse** to browse for services, or **avahi-publish** to publish them. Although the examples below use **dig**, it's purely for illustration purposes. *Avahi* tools are the way to go in real-world scenarios.

Hunting for printers

DNS-SD is a generic DNS-based mechanism to browse for network services. It's not tied to mDNS, but both are often found together. Put simply, mDNS is about naming in general, and DNS-SD is about naming (and locating) services. A service can be almost anything: a workstation, a file server, a printer, or even an SSH instance running at some Linux host.

Imagine you have a network printer. When you start it, the printer claims ownership over some unique mDNS hostname, but also announces shared a PTR record for a specifically crafted name, **_printer._tcp.local.**`The **._tcp** means that the printer is available over TCP. Anything else would use **_udp**, regardless of the actual transport protocol. **_printer** advertises LPR



Browsing for services on **local network**:

Location	Name
wlan0 IPv6	y550p
wlan0 IPv4	y550p

Avahi includes graphical tools to browse networks for various services, including SSH.

printing support. Underscores aren't permitted in DNS, so they are introduced to prevent name clashes.

This PTR record resolves to **instance._printer._tcp.local**. The **instance** part is end-user visible, so it should be descriptive. An SRV record must exist for this name, providing both host and port the service is available. Optionally, the name may also have a TXT record, containing additional connection data. This is useful for legacy protocols, like LPR.

Let's try it in action. First, I'll run **dig** to see which printers are available in the vicinity:

```
$ dig @224.0.0.251 -p 5353 -t ptr _printer._tcp.local.
;; ANSWER SECTION:
_printer._tcp.local.   10    IN    PTR    EPSON\032WF-
2010\032Series._printer._tcp.local
```

-t ptr tells **dig** that I want a PTR record; **-p** specifies the port. Now we know there is at least one printer nearby. There could be more, but **dig** doesn't speak mDNS well enough to find them all. Still, I can use it to query for the printer's details:

```
$ dig @224.0.0.251 -p 5353 -t any 'EPSON WF-2010
Series._printer._tcp.local.'
;; ANSWER SECTION:
EPSON\032WF-2010\032Series._printer._tcp.local. 10 IN
SRV 0 0 515 EPSON4E85C9.local.
EPSON\032WF-2010\032Series._printer._tcp.local. 10 IN
TXT "txtvers=1" "priority=50" "rp=auto" ...
;; ADDITIONAL SECTION:
EPSON4E85C9.local.    10    IN    A     192.168.101.158
```

In mDNS, the **ANY** record type yields all records for the given name. As you might expect, there are two. The SRV record says that the printer is at **EPSON4E85C9.local.**, port 515. Two zeroes are the service's priority and weight. The TXT record contains many key-value pairs I omitted for brevity. Among them, **rp** stores the queue name for the printer driver to use. An mDNS responder on the printer was also kind enough to include an **A** record in the **Additional** section and save us an extra DNS query.

That's it! Now we can open a TCP connection to 192.168.101.158:515 and submit a printer job. No end-user setup was necessary. Also note how the instance name differs from abbreviations like **prn120** that you usually encounter in a "big" DNS. Zero-configuration magic in action!

/DEV/RANDOM/

Final thoughts, musings and reflections

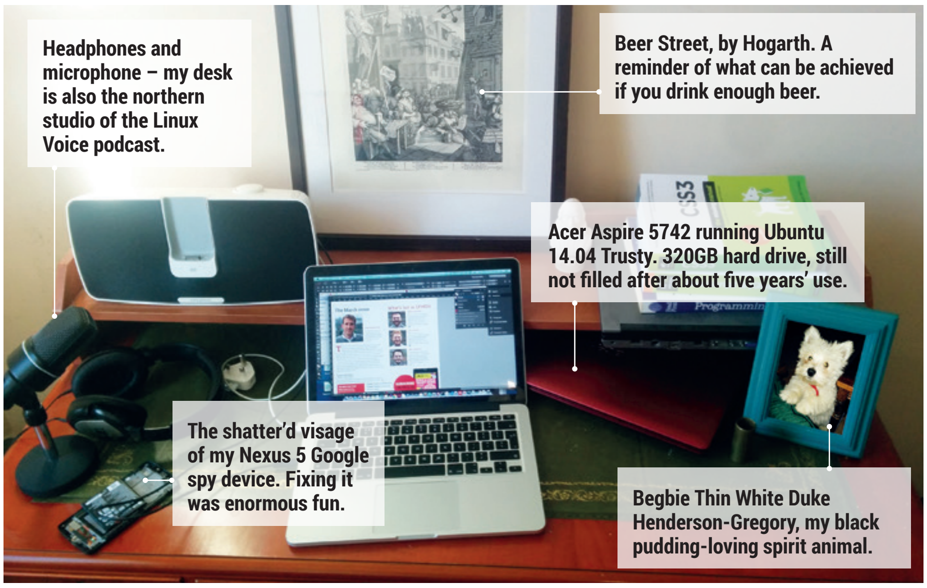
Nick Veitch was the original editor of Linux Format, a role he played until he got bored and went to work at Canonical instead. Splitter!

I remember an acquaintance of mine telling me about his unfortunate experience with a sort-of-smart car. It was the old story of the "service me now" light coming on, and being ignored to the point where, once stopped for a rest-break, the vehicle decided it was no longer going to move until it had been properly seen to by a qualified engineer with the magic box necessary to talk to the car into resuming normal service.

Sadly in this case, the rest-stop was just the side of the road in the middle of a South African desert, a good few miles from the nearest qualified engineer, any engineer, or possibly any other person at all.

Smart cars should be smart, not just at leveraging profit for their overlords. Maybe in the future when we can talk to our vehicles, we may be able to talk them out of being so stubborn. In the meantime, one of the best ways to ensure that the ever-more computerised cars of the future aren't crippled by DRM-like lock-in, crippleware and worse is to promote the idea of open source and open standards. Not everyone is looking forward to the day when you have to run a virus-check or watch a sponsored ad before the school run in the morning.

It is gratifying then that at least some collaborative efforts are taking place in this arena. The Linux Foundation has its Automotive Grade Linux project, which seems to have the backing of many players in this space, including several big-name manufacturers. you may get some idea of what the commute of the Tux-based future might look like here: **www.automotivelinux.org**



Headphones and microphone – my desk is also the northern studio of the Linux Voice podcast.

Beer Street, by Hogarth. A reminder of what can be achieved if you drink enough beer.

Acer Aspire 5742 running Ubuntu 14.04 Trusty. 320GB hard drive, still not filled after about five years' use.

The shatter'd visage of my Nexus 5 Google spy device. Fixing it was enormous fun.

Begbie Thin White Duke Henderson-Gregory, my black pudding-loving spirit animal.

MY LINUX SETUP REV. ANDREW J GREGORY

Podcaster, editorial director, permanent Linux newbie.

Q What version of Linux are you currently using?

A These days I flit back and forth between Ubuntu and Mint, and at the moment it's Ubuntu 14.04. It's an old version because I fear change.

Q And what desktop are you using at the moment?

A That would be Unity. It has its faults, and I really should get round to switching to Mate, which has become the future of the Linux desktop by sticking with and refining all the things that were good about desktops in the past – namely, it's simple to use and looks good.

Q What was the first Linux setup you ever used?

A Oooh, Mandrake something or other, in 2005. Well, I say used – there was a graphics incompatibility, so I tried a few others until I got SUSE working. I stuck with SLED until Novell signed a patent partnership with Microsoft in late 2006, which led me to Debian, then to Ubuntu. You've got to vote with your feet sometimes.

Q What Free Software/open source can't you live without?

A LibreOffice. It's amazing that this brilliantly useful software is available to us all for free. And it's not so much software, but the lack of viruses means that I can eke out hardware for longer than I would otherwise, which saves me £££s.

Q What do other people love but you can't get on with?

A I only ever use a text editor to tweak configuration files now and then, so the continuing saga of Vim vs Emacs leaves me baffled. Gedit does the job just fine, (or Sublime Text if you're feeling fancy).

PIMORONI

http://pimoroni.com http://pimoroni.de

For all things Pi <3

Join the pirates!

http://pimoroni.com/invest

#CapitalAtRisk #InvestAware

We also need to tell you: Investments of this nature carry risks as well as potential rewards. Please #InvestAware as your capital is at risk.

Approved as a financial promotion by Crowdcube Ventures Ltd, which is authorised & regulated by the Financial Conduct Authority (No. 572026).

I LOVE FREE SOFTWARE DAY

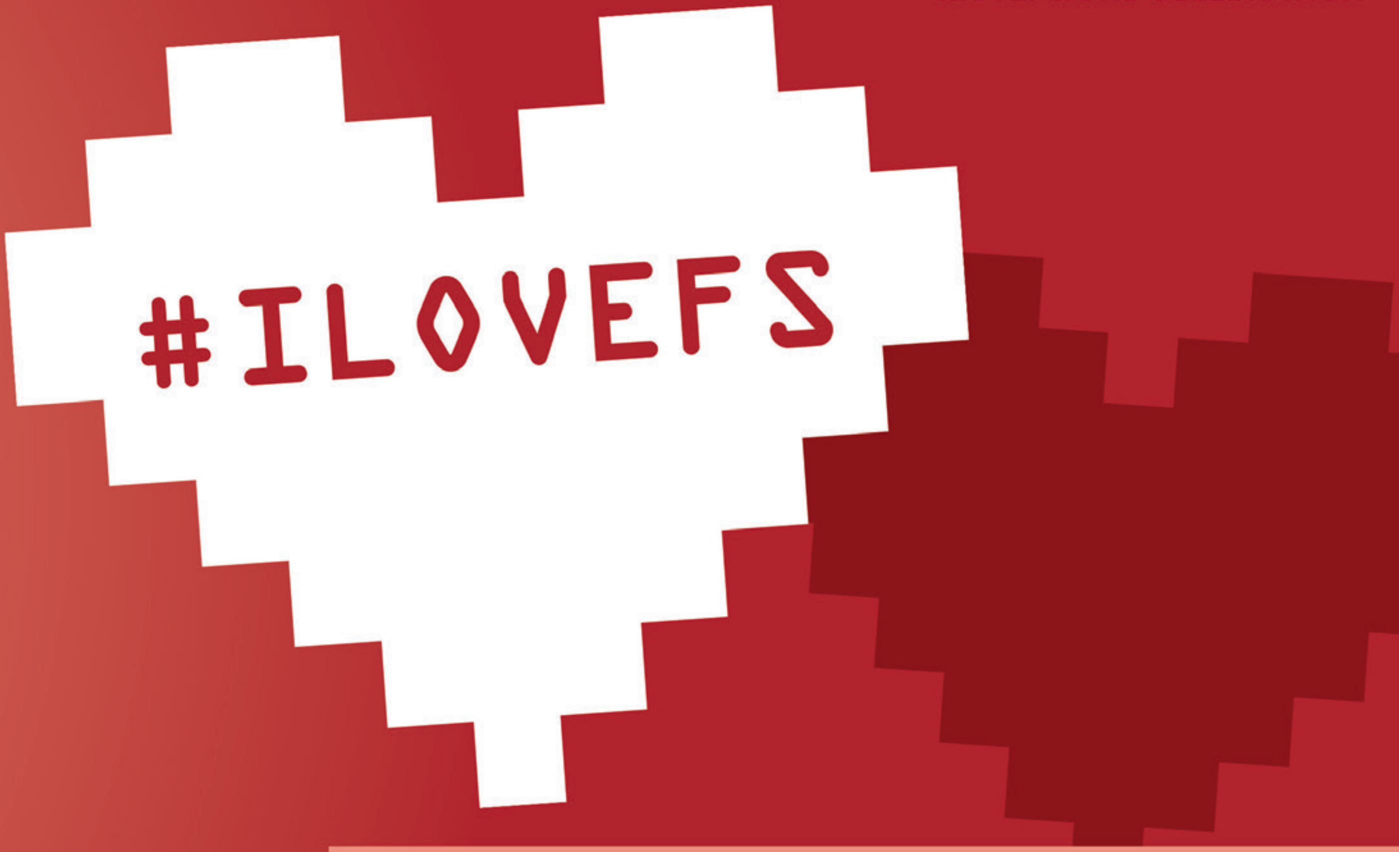
14.02. Show your love for Free Software!

In the Free Software community ...

we exchange a lot of criticism. We write bug reports, tell others how they can improve the software, ask them for new features, and generally are not shy about criticising others. There is nothing wrong about that. It helps us to constantly improve. But sometimes we forget to show the hardworking people behind the software our appreciation. We should not underestimate the power of a simple "thank you" to motivate Free Software contributors in their important work for society. So say thank you on 14th February!

fsfe.org

ILOVEFS.ORG CELEBRATION



Previously people did:

- thank contributors on microblogs, social networks, on mailing lists, and blogs
- post pictures expressing their love for Free Software
- send postcards to developers they adore
- write poems and create comics for #ilovefs
- bake muffins for their Free Software colleagues
- donate to developers and organisations dedicated to Free Software
- express their appreciation in other creative ways

Find out more:
ilovefs.org

Now what will you do on 14.02.?