FREE 222-PAGE TUTORIAL MEGAPACK DOWNLOAD

See p5 for details

LINUXVOICE

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

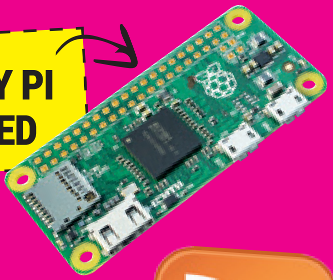
RASPBERRY PI

GPIO Zero – hassle-free hardware hacking

BUILD A LINUX SMART HOME

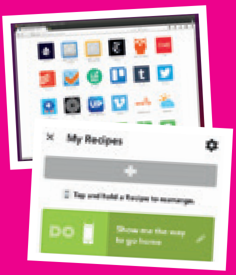
Connect mundane household items to the wonders of the internet – just because we can!

$5 RASPBERRY PI ZERO: REVIEWED



RSS
Take back your news from Twitter's proprietary clutches

IF THIS, THEN THAT
Simple scripts to save time, work better and make your life easier



PRIVACY: SECRETS OF TOR'S BROWSER

32 PAGES OF TUTORIALS

DEFENDER OF THE GPL
BRADLEY KUHN
Opportunistic GPL abusers, beware – the Software Freedom Conservancy's coming to get you...

SUSE CONF
SUSE
There's something happening at SUSE – the forgotten distro is fighting back!

MINSKY › SYNTHESIZERS › OGGCAMP & MORE!

FREE SOFTWARE | FREE SPEECH
ISSN 2054-3778
February 2016 £5.99 Printed in the UK
9 772054 377001 02

PUT LINUX IN EVERYTHING

The February 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.

It's difficult to come up with snappy terms for technology. Cloud doesn't do it for me, and neither does 'Internet of Things', which I find particularly bereft of meaning. But both technologies are incredibly exciting, and I've been hooked on the idea of home automation for some time. With the launch of the new £4 Raspberry Pi, I can envisage fitting this small, light, fully functional Linux computer into all kinds of things, maybe even the toaster.

Doing this yourself is not only an adventure, it ensures you're not beholden to whomever owns the technology. We can easily envisage a time when Philips 'upgrades' its remote lighting protocol, for example, forcing us all to upgrade too, and that's before asking where the data is going and who has access to your central heating schedule. Once again, Linux and open source is the only option.

Graham Morrison
Editor, Linux Voice

What's hot in LV#023

ANDREW GREGORY

Studying economics usually involves lots of charts and axis intersections. But being able to visualise and play with these via *Minsky* has been truly revealing.
p90

BEN EVERARD

None of us would be here without the GPL, and yet very few people defend its use and fight for our rights. Bradley Kuhn is one of those people, and our interview this month is enlightening.
p34

MIKE SAUNDERS

Ben's exceptional games tutorial, using nothing more than SVG and a few lines of JavaScript, is brilliant. And the game itself is really, really addictive. It's like *Flappy Birds* meets *Scramble*.
p88

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, Andrew Conway, Marco Fioretti, Vincent Mealing, Simon Phipps, Les Pounder, Valentine Sinitsyn.

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

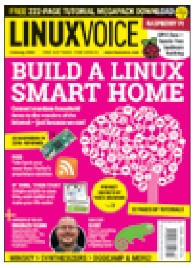
1 At the end of each financial year we'll give 50% of our profits to a selection of organisations that support free software, decided by a vote among our readers (that's you).

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.

SUBSCRIBE ON PAGE 56

WIN £4,250 OF HACKER SWAG: DETAILS INSIDE
LINUXVOICE
USE LINUX SMARTER
INSIDE: 8 REASONS LIBREOFFICE ROCKS
FIREFOX: DISCOVER ITS HIDDEN FEATURES
FREEDOM!
Switch to 100% Free Software today
9 REASONS LINUX BE
BEST LI
DESK



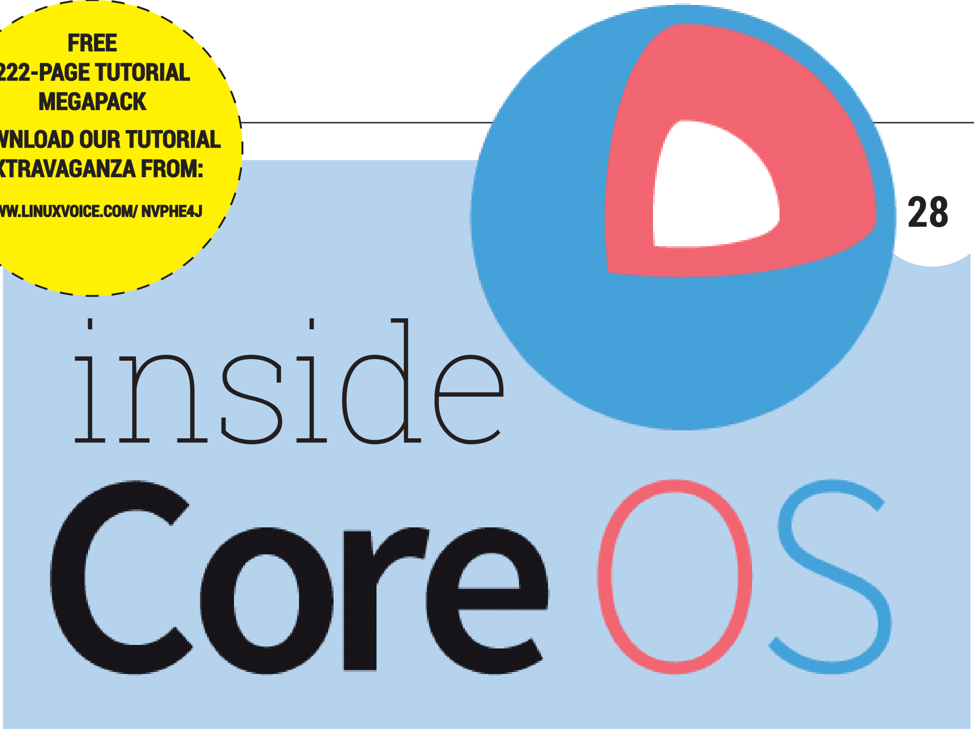
Contents

What rough beast, its hour come round at last, slouches to the sofa for a nap?

SUBSCRIBE ON PAGE 56

OGGCAMP 2015 TURN TO PAGE 14!

FREE 222-PAGE TUTORIAL MEGAPACK
DOWNLOAD OUR TUTORIAL EXTRAVAGANZA FROM:
WWW.LINUXVOICE.COM/ NVPHE4J



NEWSANALYSIS

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

Opinion

Does Microsoft Love Linux Yet?

The company that once compared Free Software with drug dealers is cleaning up its act. Or is it?

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.

It's up to you, dear readers, to make up your minds how much you trust our old friends at Microsoft.

When news broke in the Autumn that Microsoft and Red Hat had reached an agreement for Red Hat Enterprise Linux and the rest of Red Hat's software portfolio to be sold through Microsoft's Azure cloud service, many saw it as an end to hostilities from Microsoft. Indeed, anywhere Microsoft's Azure team goes, it displays the slogan "Microsoft Loves Linux" and embraces the geek subculture, right down to Microsoft-branded Tux plushies. Even this esteemed magazine carried an interview with a key Microsoft spokesman in a recent issue.

So is it true? Does the company whose former CEO described Linux as a cancer really love Free Software now? Has Microsoft joined the FOSS community?

The answer has to be "not yet". The Azure cloud business unit has definitely understood that it can only succeed by embracing Linux, and its influence has led to other projects (such as the .Net developer tools Azure needs) also taking open source steps. It would be fair for Microsoft to use the slogan "Azure Loves Linux".

But the rest of the company is not so advanced. As far as anyone can tell, the Windows and *Office* business units are still solidly proprietary, with the tiniest of concessions to their closed software working on open platforms being treated as salvation. Worse, there seems to be no let-up in their continued taxation of open source community members using embedded Linux or adding compatibility with Windows filesystems, and revenues from enforcing software patents against Android appear to be at an all-time high.

According to personal reports I've received, they also still seem to covertly spread FUD about open source applications here in Europe when there is a risk of regional governments adopting ODF or *LibreOffice*. So while the Azure business unit is embracing the ecosystem the same as many before them have done, the Windows and *Office* business units show no signs of "loving" Linux, and only modest signs of co-existing with open source.

> Microsoft still seems to spread FUD about open source when there is a risk of regional governments adopting ODF or LibreOffice

It's hard to change a company as large and profitable as Microsoft quickly. But a significant and binding gesture of goodwill would go a long way to convincing those of us with the scars of Microsoft's decades of verbal and actual abuse of open source that they mean business. One aspect of the Red Hat press release was its mention of a "patent stand-still agreement" with Microsoft. Clearly Red Hat is wary of Microsoft's ongoing patent aggression, and if they are concerned, so should we be.

That's why my recommended remedy for Microsoft concerns patents. If Microsoft really wants to convince us all that it has changed and now wants to be considered a full member of the open source community, it needs to foreswear patent aggression. It could do that by joining the Open Invention Network (OIN) or by signing up to Mozilla's Open Software Patent Licence. But until Microsoft ceases hostilities, it's too early to say that the whole company loves Linux or open source.

Tor Messenger • LibreOffice • Hurd + Gimp • Kernel • Jolla • Debian

CATCHUP

Summarised: the biggest news stories from the last month

1 Tor-based Messenger chat program hits beta

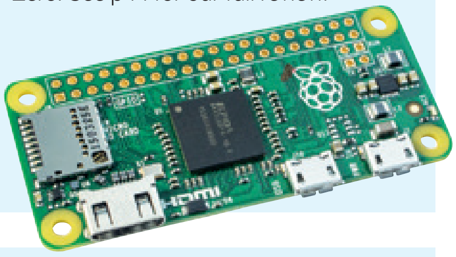
Tor Browser has become the go-to program for anonymously accessing the web, and now *Tor Messenger* is here to do the same for instant messaging. The program supports multiple transport networks such as Jabber, IRC, Google Talk and others, and enables OTR (Off-the -Record) encryption and authentication by default. Downloads are available for Linux, Windows and Mac OS X. **https://blog.torproject.org/blog/tor-messenger-beta-chat-over-tor-easily**

2 LibreOffice team holds bug hunting session

LibreOffice 5.1 is undergoing heavy development work, and the team is trying to get more coders involved by holding bug hunting sessions. Experienced *LibreOffice* hackers will be available to assist and mentor newbies who want to fix bugs in the upcoming 5.1 release, but don't know their way around the labyrinthine codebase. All being well, *LibreOffice 5.1* final will be released in early February 2016, and promises to start up twice as quickly as the current stable release.

3 New Raspberry Pi Zero costs less than a pie

After admitting he'd spent his entire savings twice when he was younger, first on an Acorn BBC then on a Commodore Amiga, Raspberry Pi CEO, Eben Upton, launched the incredibly affordable £4 Raspberry Pi Zero. See p44 for our full review.



4 GNU Hurd 0.7 released

When Richard Stallman started the GNU project, he didn't expect the Linux kernel to arrive several years later and take a lot of the Free Software fanfare. GNU has been working on its own kernel, Hurd, although development has been painfully slow. Being a microkernel, where drivers, filesystems and network stacks are kept in separate processes, Hurd could one day be even more reliable than Linux (but with significant performance impacts). **http://tinyurl.com/pjg3zae**

5 Gimp turns 20 with 2.8.16 and shiny new website

Yes, the GNU Image Manipulation Program has turned 20 years old. Developer Peter Mattis announced the first publically available snapshot of *Gimp* to the **comp.os.linux.development.apps newsgroup** on 21 November 1995. Two decades later, the project has undergone a (long needed) website revamp, and a new *Gimp* release – 2.8.16 – is here with support for layer groups in OpenRaster files, PSD fixes and other updates. **www.gimp.org**

6 Linux kernel 4.4 to bring Pi graphics driver + more

Version 4.4 of the Linux kernel should be available by the time you read this, and brings about a boatload of improvements and updates. A KMS (kernel mode setting) driver for the Raspberry Pi has been included – although hardware acceleration and power management will come in a later release. Linux 4.4 also includes support for AMD's Stoney APU, along with VirtIO VirtGL which provides OpenGL support for guest virtual machines running inside the *Qemu+KVM* combo.

7 Jolla lays off staff, goes for debt restructuring

Bad times for Jolla, developers of the smartphone of the same name and the Linux-based Sailfish OS. The company has temporarily laid off most of its staff and applied for debt restructuring, following the withdrawal of its largest investor. "Jolla is now fighting for survival", says co-founder Antti Saarnio, and the news raises big questions about Jolla's planned tablet. More details here: **https://blog.jolla.com/open-letter-jolla-community**

8 Debian Live falls victim to developer spat

"Debian can be great. But depending on who you are, where you come from, and who your friends are, Debian can also be hateful and full of deceit." So says Daniel Baumann, lead developer of Debian Live, claiming that his work has been hijacked by other developers inside the Debian-CD and Debian-Installer teams. Those teams claim it was necessary in order to add new features such as UEFI support and fix long-standing bugs. **http://tinyurl.com/qdyjq4x**

DISTROHOPPER

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

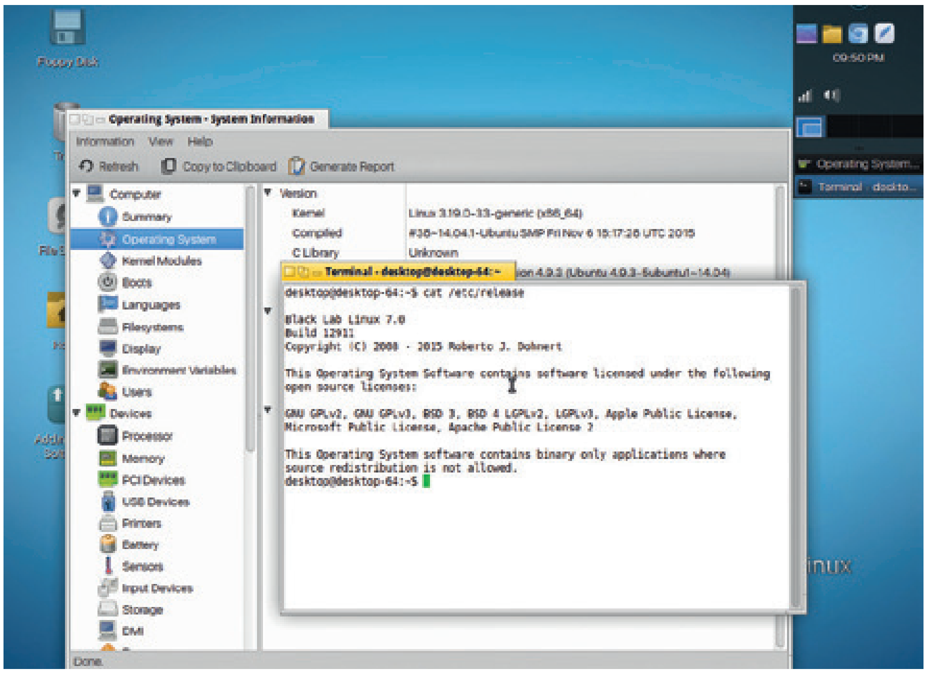
Black Lab Linux 7.0

With commercial support.

New Linux convertees often have a hard time. They can be bamboozled by the sheer number of distros, desktop environments and window managers, and when they finally settle on something and get it installed, finding help can be a whole other challenge. "RTFM" is the usual response ladled out by many in the community – while the friendlier among us try to point newbies in the direction of forums, mailing lists and wikis.

Black Lab Linux aims to fix this problem by offering a polished home desktop distro with personal support. It's available in two versions: for $89.99 USD you get one year of phone and email support, while for $25.99 you get 30 days of email support. The former may seem like a lot of cash, especially when many of us are used to fixing problems ourselves for free, but the ability to call someone at any time and get personalised help could be a major attraction for many potential Linux newbies.

Black Lab Linux 7.0 features an Xfce desktop with LibreOffice 5, Chromium (including the Pepper Flash plugin) and Steam Client, all running on kernel 3.19. Readers who recall the days of BeOS will notice a lot of similarities in the desktop here, especially those title bars... Ah, nostalgia. **www.blacklablinux.org**



```
desktop@desktop-64:~$ cat /etc/release

Black Lab Linux 7.0
Build 12911
Copyright (C) 2008 - 2015 Roberto J. Dohnert

This Operating System Software contains software licensed under the following
open source licenses:

GNU GPLv2, GNU GPLv3, BSD 3, BSD 4 LGPLv2, LGPLv3, Apple Public License,
Microsoft Public License, Apache Public License 2

This Operating System software contains binary only applications where
source redistribution is not allowed.
desktop@desktop-64:~$
```

Got a friend who wants to try Linux, but needs help getting started? Black Lab could be the answer.

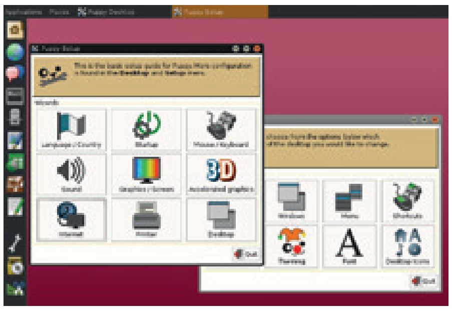
Puppy Linux 6.3.0

Slackware goodness fine-tuned for older machines.

We're often asked which distro is best suited for an old PC. Of course, that depends on the definition of "old", but in most cases we recommend Lubuntu for machines with 1GB+ of RAM, while Puppy Linux is a great alternative for boxes with 512MB.

Puppy Linux 6.3.0, the latest release, is based on Slackware 14.1, which in itself is a fairly minimalist distro that uses mostly vanilla upstream sources. Puppy is designed to be compatible with Slackware's package repositories, so you get access to hundreds of apps and utilities. Puppy itself is a very trimmed-down distro that uses the *JWM* window manager alongside the *ROX* file manager as its desktop setup.

Almost all of the software supplied with Puppy is friendly to your RAM banks as well: there's the *Sylpheed* email client, *AbiWord* word processor, *mtPaint* image editor and other tools. We were surprised to see *Firefox* included as the default browser – sure, it's a great browser, but *Midori* or *NetSurf* (see our Group Test on page 50) would have been more appropriate alternatives.



Got an old PC or laptop sitting around doing nothing? Give it a new lease of life with Puppy.

Puppy has plenty of fans on its forums ready to help newbies, and development is active so you don't have to hunt down a distro from five years ago just to find something that runs decently on an old PC.

News from the *BSD camps

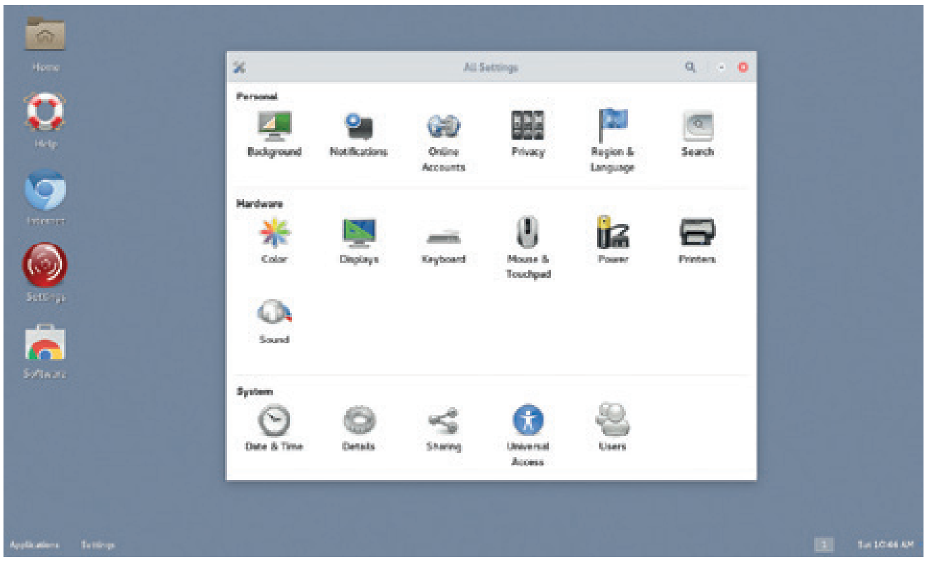
What's going on in the world of FreeBSD, NetBSD and OpenBSD.

OpenBSD head honcho Theo de Raadt often gets flak for his no-nonsense attitude on mailing lists, and some developers in other projects have criticised him for being "arrogant" and out of touch. A video interview from a BSD conference in 2013 (**www.youtube.com/watch?v=OXS8ljif9b8**) shows another side to the man, however; in it he freely admits that OpenBSD has historically been weak in some areas, most notably in support for SMP (multiple processors) and virtual machines.

The latter deficiency is being fixed though: developer Mike Larkin has started to introduce code for the new VMM subsystem into the main source tree. Currently, VMM is a simple hypervisor that makes it possible to run OpenBSD guest virtual machines on an OpenBSD host, but due to a lack of a BIOS there's no proper support for other operating systems right now. (Larkin noted that he has managed to get a NetBSD kernel booting with a bit of hackery.)

The VMM code is undergoing heavy development, and as stated on the mailing list, "There are still lots of things that need to be fixed and improved. For this reason, it is still disabled by default." OpenBSD has traditionally been very conservative with new features, focusing intensely on code quality and security rather than having every bell and whistle available from day one, so we're intrigued to see how VMM develops.

Meanwhile, if you've always wanted to get into the world of BSD but found the installation processes rather taxing, try DesktopBSD (**www.desktopbsd.net**). It's a respin of FreeBSD with new users in mind, and to that end it has a graphical installer and other tools to make life easier for first-timer BSDers. The DesktopBSD team has just announced the first milestone release towards version 2.0, so grab it from the site for testing.



DesktopBSD takes a mature and well-tested FreeBSD base and peps it up with a graphical installer and other newbie-friendly goodies.

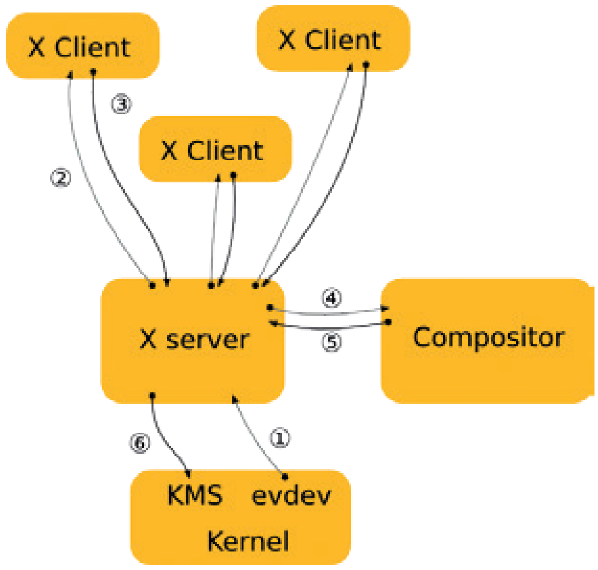
Wayland: finally here?

It feels as if Wayland, the replacement for the venerable X Window System, has been "just around the corner" for many years now. For a lot of people in the GNU/Linux community, there's nothing wrong with X – so if it ain't broke, why fix it? X has served us well over the years (no pun intended), providing some features like network transparency that have been hugely useful for setting up thin-client desktop terminals.

But X is still full of cruft and hacks that have built up over the decades. It often takes a lot of fiddling to get good performance from the X server, eg without tearing effects when dragging windows around, along with other issues, so we need something that's simpler, more streamlined, and more usable across different devices including smartphones and tablets.

Wayland has already seen some (limited) real-world use in the Sailfish OS running on the Jolla smartphone – although the future of Jolla is looking shaky now, as reported on page 7. In regular distro-land, we've had the bizarre RebeccaBlackOS as a testing ground for Wayland, but none of the big-name distros have adopted it yet.

That's about to change though: Fedora 24, due in May 2016, will likely have Gnome running on Wayland as the default desktop. Fedora 23 uses Wayland for the login screen, but this switch for the complete desktop will be a huge – and possibly risky – change. Still, somebody has to do it, so even if Fedora 24 ends up being released with some bugs and glitches on the desktop, hopefully this will lead to widespread usage and testing of Wayland, so that the community can iron out the wrinkles and prepare for an X-free future.



Wayland aims to replace the X Window System's legacy architecture with something simpler and more streamlined.

YOUR LETTERS

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

STAR LETTER

TALKING TO PEOPLE ABOUT PRIVACY

I listened to the 5 November podcast today., and I also believe that the general public needs to be made aware of privacy issues, and I have also encountered people who don't initially seem concerned about it. To discuss the issue with the ignorant masses, you have to put it in a form they understand. When I encounter a person who says they aren't concerned, I follow this "program":

```
Ask the person for their wallet.
If they don't offer it
Ask why they didn't offer it.
If the reason is that they don't want anybody to see what's in it
Open the discussion into internet privacy.
Else open the discussion into internet privacy.
Else
Open their wallet.
Until the wallet is empty or the person protests  # Protest is an immediate interrupt to this loop
Take an item from the wallet.
If you're in a room full of people
Read the information from the item aloud to the room.
Else copy the information to a piece of paper.
Give the person back their wallet and items.
If the person protested the release of their information
Open the discussion into internet privacy.
Else if there was nothing of importance in their wallet
Congratulate them on the mugging decoy.
Open the discussion into internet privacy.
Else you're dealing with somebody who has no concept of the idea of privacy.  Explain it to them, then enter into the discussion on internet privacy.
```

Paul Olson, Eufaula, Oklahoma, USA

The writer of the dreadful Telegraph piece linked to below is an advisor to this man. On unicorns, probably.

Andrew says: To misquote WB Yeats: The best lack all conviction, while the ignorant spew rubbish without shame because they don't realise just how daft they sound. Unfortunately the great mass of comment about privacy tends towards drivel (as with this idiocy: **www.telegraph.co.uk/technology/12008689/Why-is-Silicon-Valley-helping-the-tech-savvy-jihadists.html**). Thanks very much for this clear sighted good sense.

DEBIAN!= LINUX

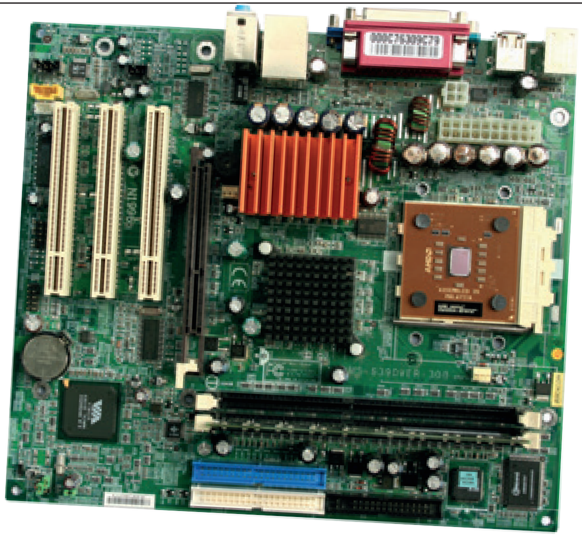
SUSE won us over years ago for its superb hardware detection, but it's pushing ahead in all sorts of new areas now.

I was an OpenSUSE desktop user for a few years before switching over to Mint. I wanted to install Steam, which really works smoother on Ubuntu-based distros. This helped me to realise some of the differences in userland. I'd like to comment on the letter titled 'Cinnamon' in Linux Voice 21. It would appear John doesn't realise that sudo is a Debian, or perhaps Ubuntu thing. Mageia, like Mandriva before it, like Slackware, OpenSUSE and I believe Red Hat use "su" as the default method to execute commands as root. You can, of course, set up sudo, but it's not standard, nor necessary. I find it interesting, and I suppose a little sad that people become so familiar with commands that are specific to one distro, or family of distros, that they take it for granted that there are other ways to do things. Another nasty surprise for me was when my script, which used 'rename' in SUSE didn't work in Mint. A quick look at both man pages tells all: same command name, but they don't work the same. So in Mageia's defence, you don't have to include an extra and unnecessary command to be 'complete enough'.
Bruce Muggins

TINKER, TAILOR

Following Lenovo's foray into spying, by placing software on its desktop and laptop motherboards to gather information not related to making the computer work; would the Linux Voice team like to consider how long it will be before Cameron and Obama take a deeper walk into 'George Orwell land', and finance computer manufacturers to purpose-design motherboards to harbour spying software that does not need to load onto a hard drive. This would appear to be the next logical step in monitoring everything that we do on our computers. How possible or probable this is? Woud this compromise the Tor Browser? Would such a system be able to spy on our encryption codes?
John Bourne

Andrew says: I'd be very surprised if Chinese-made models were completely government spyware-free. Any western company would be more interested in the short-term interests of its shareholders than the government of the day though; if it were to come out that, say, HP were installing government spyware on its models, it would expose itself to a ton of legal action and would lose a ton of money. That's more important than anything else.

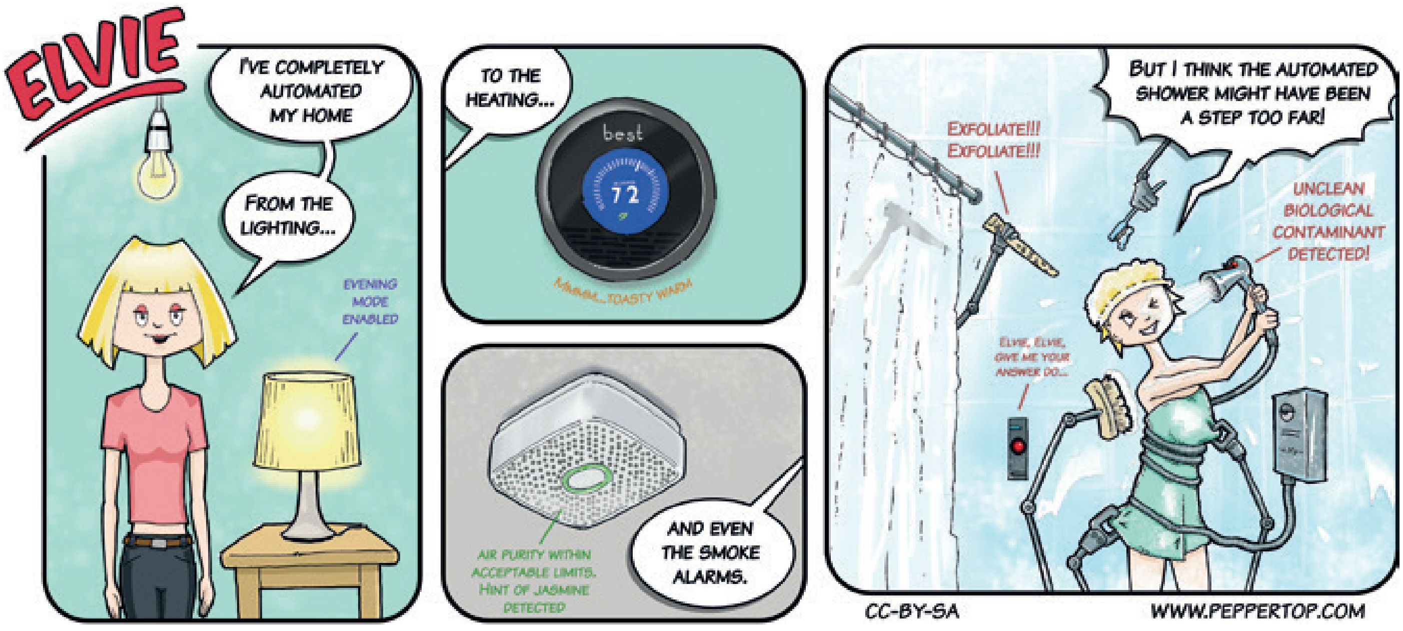

There's no way to tell whether this motherboard is infected with any malware. Be careful!

PENDANTS CORNER

I am a subscriber and regular reader of Linux Voice since its beginning. I have been working with computers since January 1968. My wife, a retired primary school headmistress, uses a Lenovo laptop bought through Amazon which came delivered with Linux installed and with the username defined. Just the password needed to be changed. Since then we have upgraded her to Linux Mint Mate 17.

On glancing at the cover of the January 2016 Linux Voice my wife was moved to ask "What is Linux Smarter and do we use it on our machines?" I explained that it was not a software product but that the author meant "Use Linux more smartly", "improve your use of Linux" or "use Linux in a smarter manner." She then asked whether it was an American magazine, as perhaps it was just an Americanism. Anyway, thank you for the contents, I have hidden the cover so I am allowed to read quietly with my glass of wine...
Peter Maunder

Andrew says: I've met a few Americans and they were jolly good chaps and chapesses. Their cars are rubbish, but the way they're adapting the language of their colonial overlords is pretty good, in my opinion. Apart from obligate; that's a horrid, horrid word.


ELVIE
I'VE COMPLETELY AUTOMATED MY HOME
FROM THE LIGHTING...
EVENING MODE ENABLED
TO THE HEATING...
best
72
MMMM...TOASTY WARM
AND EVEN THE SMOKE ALARMS.
AIR PURITY WITHIN ACCEPTABLE LIMITS. HINT OF JASMINE DETECTED
BUT I THINK THE AUTOMATED SHOWER MIGHT HAVE BEEN A STEP TOO FAR!
EXFOLIATE!!! EXFOLIATE!!!
UNCLEAN BIOLOGICAL CONTAMINANT DETECTED!
ELVIE, ELVIE, GIVE ME YOUR ANSWER DO...
CC-BY-SA WWW.PEPPERTOP.COM

Subscribe

shop.linuxvoice.com

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

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

US/Canada subs prices

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

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

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

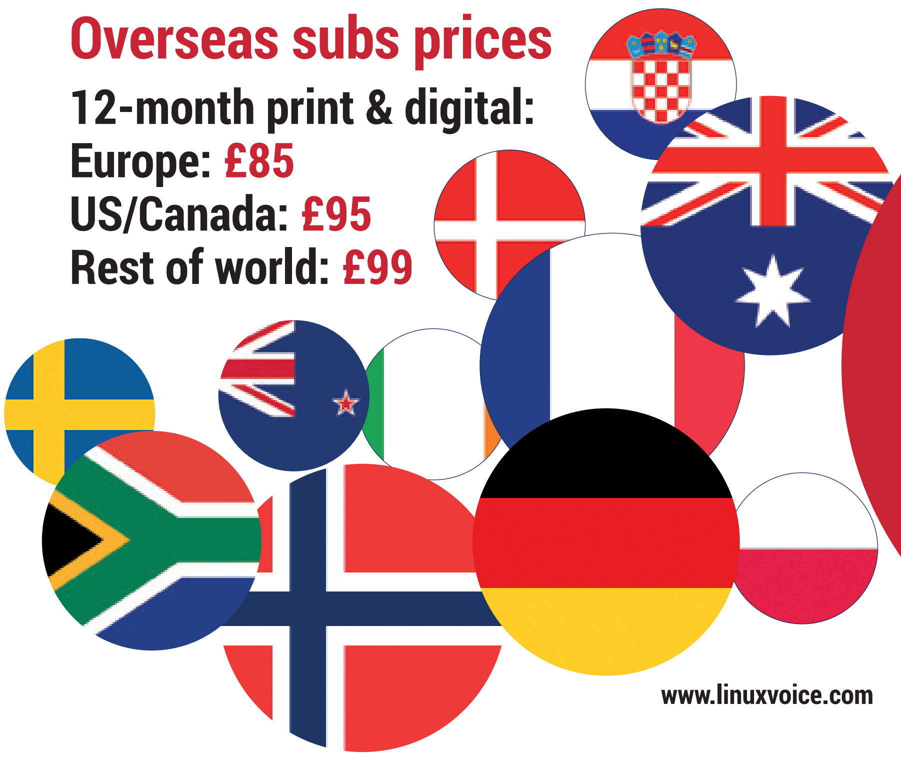
Save money on the shop price and get each issue delivered to your door

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

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

Overseas subs prices

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



DIGITAL SUBSCRIPTION*

ONLY £38

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

Graham Morrison enjoys another great year at the UK's best open source gathering [objectivity disclaimer: we sponsored the event].

Even if you're not in the UK, where OggCamp has been held since 2009, we know you'll be familiar with the atmosphere that pervades this whole weekend – it's like returning to your home town and meeting up with old school friends, many years after heading off to seek your own fortune. For us, it also turned into a bit of a road trip, as we headed north to Liverpool from our bolt holes in the South West of England. We got there eventually, and were incredibly grateful to our Airbnb host for driving us over to the Friday night drinks venue, even if that did mean a four-mile walk back in the pouring rain at three in the morning.

The venue for this year's event was the John Lennon Art & Design Building at John Moores University, literally in the shadow of Liverpool's modern Metropolitan Cathedral. This is where the event was held in 2013, before a brief sojourn to Tolkien's Oxford last year. The University buildings are well suited to an event like this because there are lots of different spaces of different sizes. There was an area for hacking hardware, for example, and another for attempting a Nerf Gun Challenge, an event successfully hacked by several children in attendance. It also meant that there were plenty of rooms with varying capacity. This was important because OggCamp is an 'unconference': that means there's no fixed schedule, and attendees propose their own talks on the day, with rooms assigned according to an event's popularity. Previous years have used an online system to manage this live scheduling, but the last few have see this break down to the point that this year we were using a whiteboard and Post-it notes.

The talks themselves were generally excellent. The weekend kicked off with Gurbir Singh talking about the new space race to Mars. If we're being completely

Held over Halloween weekend, there was plenty for attendees of all ages to do, including a Nerf Gun challenge that was successfully hacked by the children playing it.

There was cake, tea and coffee, plus two excellent evening events. We also recorded our own podcast live, which you can listen to here: **www.linuxvoice.com/podcast-season-3-episode-19**.

honest, it failed slightly to deliver on its potential. The audience was keen, and it was obvious that Gurbir is incredibly knowledgeable and enthusiastic about his subject, but we wanted to hear more than he was given time to deliver.

Talk talk

Canonical's Alan Pope, for example, gave a great presentation describing how the Ubuntu Phone app store was 'Pwned' by a malicious app that got more access than was safe. Stuart Langridge, of the excellent Bad Voltage podcast, talked about putting your podcasts on YouTube, which we've been doing since his presentation, and Jon 'The Nice Guy' Spriggs gave an excellent talk about Sandstorm – easy web containers for self hosting, which we'll hopefully be writing about for a future issue of Linux Voice. All three of those presentations are available on YouTube, although the audio quality is poor. We also want to desperately experiment with an RF interface and our central heating system, after a talk by Tim Gibbon that hacked the frequencies used by many of the controllers used in UK.

As tradition dictates, there was a great podcasters' panel (featuring our Ben!), with a long discussion about privacy, and a live recording of episode 19 of our own podcast, complete with the bells of the Metropolitan Cathedral.

Our own long-time contributor, Les Pounder, made OggCamp 2015 his last as 'The Chief' – controller of event logistics. After years at the helm, he announced he was bowing out during the final session and prize draw. The audience gave a heartfelt thank you for all his hard work, and we could see the edges of his resolve soften in the glow of such appreciation. Event organiser, founder and Linux Outlaw, Dan Lynch, also received a huge applause for his incredible efforts putting everything together, made no easier by recent ill health. Thanks Dan, and get well soon!

None of this would have been possible without the sponsors, of course, and laptop, PC and server crafters, EntroWare, deserve a huge amount of credit for being this year's Platinum Sponsor. These lads from Liverpool have built a great company behind their Linux-running hardware, and it was lovely to meet them, share a few pints and play with their Steam Machine. Both Ubuntu/Canonical and Fedora also helped sponsor the event, with stands full of stickers and hardware in the exhibition space.

We got to see Ubutu phones running the mythical convergence mode, switching between a touch and windowed environment while connected to a screen, and Jon Archer on the Fedora stand was also giving away DVDs and running the latest Fedora on a laptop – still the best way to help people get over any initial Linux anxiety.

This year's event was a huge success, and we wouldn't have expected anything less. We know it takes a huge amount of effort from a huge number of volunteers, and we're incredibly grateful to them all for giving up their time and putting so much into each day. With a bit of luck, after the hangovers become a distant memory, they'll reconvene and start planning OggCamp 2016. Fingers crossed. No Pressure.

OPEN SOURCE SMART HOME

Give your home a brain with the Internet of Things.

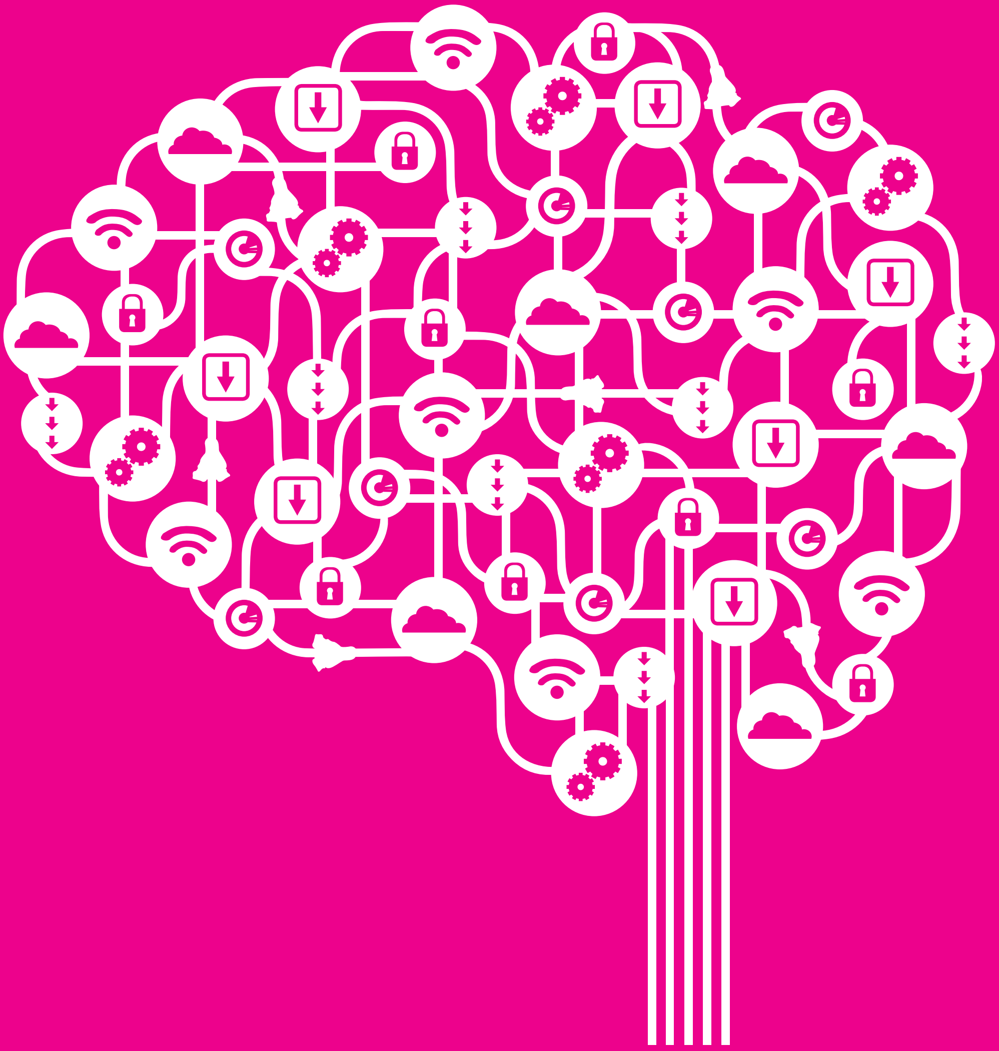
The idea of a computer-controlled home has been in science fiction almost as long as there has been science fiction, and with the advent of cheap, small, low-power modules and ubiquitous Wi-Fi smarthomes have finally become a realistic option.

Now that you can get an entire computer capable of running Linux for under £30, it's time to start putting them around the house. These devices can monitor anything from your heating to your burglar alarm, and control every aspect of your environment until you're living in computer-maintained luxury.

Want to be able to turn your heating on before you get home? No problem. Want to get an email when your burglar alarm goes off? Easy. Want a robotic arm to hand you an umbrella when you open the door and there's rain forecast? That'll take a little more work, but it is possible.

Your freedom matters!

The same principles that make open source essential to having control of your computer mean that open source is also vital when it comes to having control of your home: the more power your devices have, the more control we as users and developers should have to make sure we can run them as we want to. What's more, your smart home will monitor details about your life that you may want to keep off the internet. Fortunately, there are plenty of tools that we can use to build a powerful home setup without compromising our software freedom.



The more power your devices have, the more control we as users should have to make sure we can run them as we want to

BUILDING THE **BRAIN**

Create the hub that will control your house.

Our smart home will be built around a hub to manage the data and control the peripheral devices. This hub will provide a simple user interface to visualise the data and enable us to control our home. It's perfectly possible to build this hub from scratch using any programming language of your choice – however, there's no need to go to these lengths, as there's an open source framework for smart homes already available: *OpenHAB*.

OpenHAB is written in Java so it's platform-independent. You can use it on just about any device that's capable of running Linux (or Mac OS X or Windows if you prefer). We used a Raspberry Pi, but any small low-power machine will work well whether it's ARM- or x86-based. Since we're using a Raspberry Pi, our instructions are for Debian-based systems, but it will be easy to convert them to other Linuxes. The only extra your device will need is a way of connecting to the network, which could be either a Wi-Fi dongle or an Ethernet cable.

The first task, as always, is to get an up-to-date version of Linux running on your hardware by following the manufacturer's instructions. Once it's installed, configure your network connection and you're ready to start. It will be easier to manage if you assign your hub a static IP on your router. Check your router's documentation for details of how to do this.

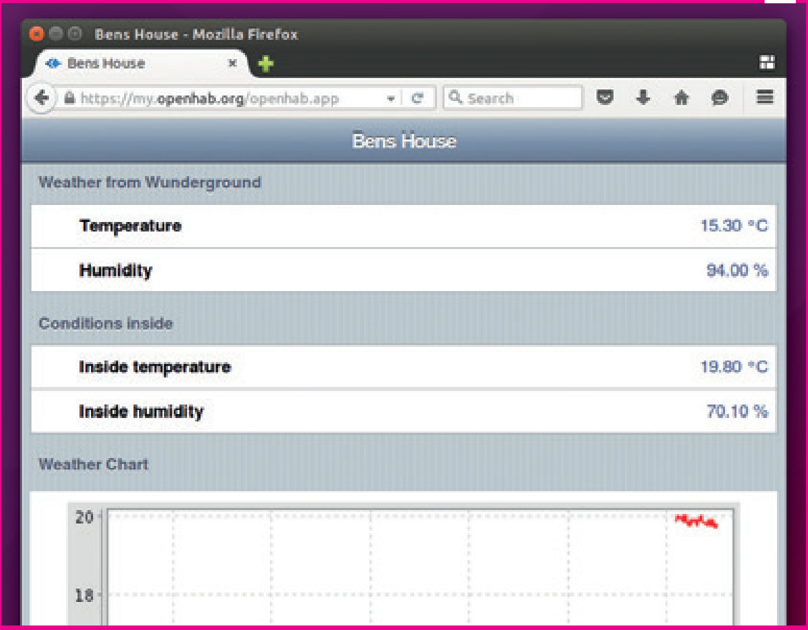
If you're using Raspbian, you should already have Java installed, but it's best to check, like so:

```
$ java -version
java version "1.8.0_66"
Java(TM) SE Runtime Environment (build 1.8.0_66-
b17)
```

Since stability is important on software that controls your house, we're inclined to stick with the officially supported Oracle Java rather than OpenJDK.

Download the *OpenHAB* **Runtime Core** and **Addons** packages from **http://www.openhab.org/getting-started/downloads.html**. These come as Zip packages, but lots of the files are in the root of the Zip, so it's best to put them in their own directories before unzipping them:

```
mkdir openhab
cp <runtime-core> openhab/
cd openhab/
unzip <runtime-core>
```


Bens House - Mozilla Firefox
Bens House
https://my.openhab.org/openhab.app
Search
Bens House
Weather from Wunderground
Conditions inside
Weather Chart
20
18

Temperature	15.30 °C
Humidity	94.00 %
Inside temperature	19.80 °C
Inside humidity	70.10 %

The *OpenHAB* web interface is simple, yet you can still control just about anything with a few clicks.

OpenHAB needs some quite detailed setup before it becomes useful. This means it's quite slow to get it running for the first time, but once this is done, you'll have exactly the system you want.

Personalisation

Configuration is mostly centred around items and sitemaps. As you can probably guess from the names, items are individual things that are either data coming in, or services that *OpenHAB* can trigger. Sitemaps describe the layout of these items on the final user interface (which can either be presented through a website or an app for mobile phones). The exact configuration of items and sitemaps will depend on the hardware you have running in your smart home, but first let's look at a simple setup that doesn't depend on any external hardware: a weather forecast. This can be important to your final setup because you may choose to run your home slightly differently depending on the conditions outside. This could be anything from turning on a sign reminding you to take an umbrella if there's rain expected to fine-tuning your heating setup.

The weather forecast comes from an *OpenHAB* addon, so you need to extract the **addons** Zip file downloaded earlier. We'll need a few different

Competing standards

There are a few different protocols used in commercial products. Here are a few of the most popular:

- **Lightwave** Accessible to *OpenHAB* through a Lightwave Wi-Fi Link.
- **Z-Wave** You can plug a controller into your *OpenHAB* hub and use the Open Z-wave Control Panel to set up your network.
- **Nest** The cloud service has an API that *OpenHAB* can use.
- **Thread** Not yet compatible with *OpenHAB*.

addons in this article so you may as well copy them all over to the **addons** directory inside *OpenHAB* now. You'll need: **org.openhab.binding.mqtt-1.7.1.jar**, **org.openhab.binding.weather-1.7.1.jar** and **org.openhab.persistence.mysql-1.7.1.jar**.

This *OpenHAB* addon connects to online weather forecasts and downloads the appropriate information for your location. All the configuration of *OpenHAB* is done in the configuration folder. In there, you'll find the **openhab.cfg** file that holds all the application options. If you open this in a text editor and search for 'weather' (it's quite a big file), you should find the appropriate section:

```
############### Weather Binding #############
# The apikey for the different weather providers,
...
#weather:apikey.Wunderground=
```

You can choose to get your weather forecasts from any one of five different providers. Unfortunately OpenWeatherMap didn't work for us (the problem may be resolved by now), so we opted for Weather Underground. The process is exactly the same for each provider and it's trivial to switch between them after you're set up.

Before providing forecasts, each of the data sources requires you to sign up for an API key. This is different for each provider, so head to their website and follow their instructions.

Any line in the *OpenHAB* configuration file with a hash is ignored (commented out). You should see that all of the weather configurations are commented out by default, and you need to delete this hash in order to enable them. First, add your API key and uncomment the appropriate line; then you can move on to the location configurations.

You can define as many locations as you like (though you may be limited to the number of times you can request weather data by the forecast provider). We defined a location called **briz** that gets the data for Bristol, UK, from Weather Underground every ten minutes:

```
# location configuration, you can specify multiple ...
weather:location.briz.latitude=51.454514
weather:location.briz.longitude=-2.587910
weather:location.briz.provider=Wunderground
weather:location.briz.language=en
weather:location.briz.updateInterval=10
```

You can change the name **briz** to be anything you like, provided you change it everywhere. The only other thing you should have to alter is the latitude and longitude for your location. The **updateInterval** setting is the number of minutes between each time *OpenHAB* will request additional information from the provider, so you need to make sure that this is within your API limits.

This sets up the weather provider, but it doesn't create items that we can use. These have to be created manually in an items file. You can have as many items files as you like. The only requirements are that they end with **.items** and that they live in the **configurations/items** folder. We decided to create a new one for each different set of items we created, so the first one we called **weather.items**.

Items files contain a series of item definitions, one per line. They're in the format:

```
<item type> <item name> <label> <binding options>
```

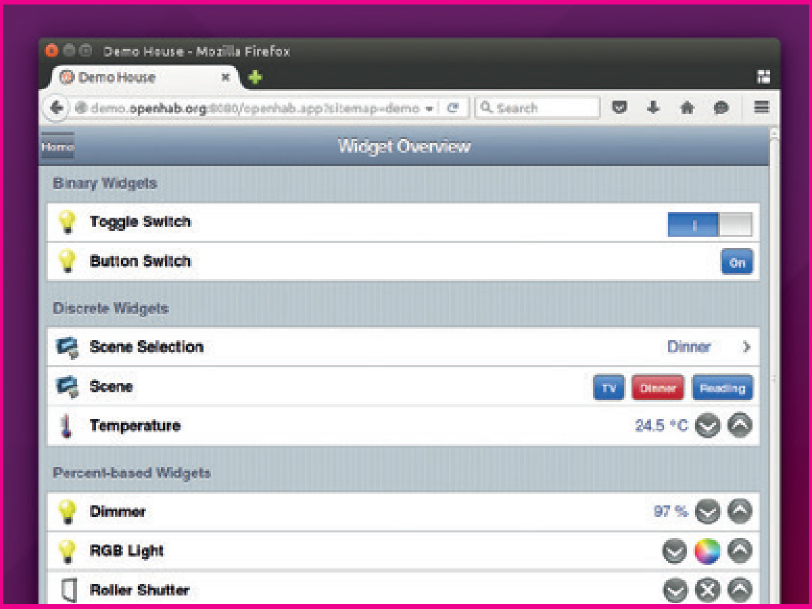
To test everything out, create items for the current temperature and humidity. These will both be of the item type **number**. The **label** is the text that's displayed alongside these items in the user interface, and it should include a place to put the actual data. The **binding options** are specific to the weather binding and include the name of the binding and the data we want to pass back. The weather items are:

```
Number   briztemp   "Temperature [%.2f °C]"
{weather="locationId=briz, type=temperature,
property=current"}
Number   brizhumid    "Humidity [%d %%]"
{weather="locationId=briz, type=atmosphere,
property=humidity"}
```

The percent character is special in the label text. **%0.2f** tells OpenHAB to output the data with a decimal point in this location; **%d** tells *OpenHAB* to use a whole number, and **%%** tells *OpenHAB* that you want a percent sign in the final output. The square brackets tell *OpenHAB* which part of the label is the actual data.

Now we have some items, we can start to build our interface to display everything. You define the user interface in one or more sitemaps, which live in the **configurations/sitmaps** directory and end with the **.sitemap** suffix. You can create as many of

The OpenHAB configuration can be a little confusing, but there's a demo house at **http://demo.openhab.org:8080/openhab.app** that you can use to copy syntax (the configuration is available in the main downloads page).



these as you like, but the one that *OpenHAB* will look at first is **default.sitemap**.

Sitemaps are hierarchical, with different sections enclosed in parentheses. The highest level always defines the sitemap itself and gives it a label. Inside this you can add items or frames that group items together. A sitemap to describe the current temperature in Bristol is:

```
sitemap default label="Bens House"{
      Frame label="Weather from Wunderground" {
           Text item=briztemp
           Text item=brizhumid
      }
}
```

The items we created to hold the temperature and humidity were numbers, but here we're defining them as text because that's how they'll appear on the screen. The item type and sitemap type aren't directly related.

Now everything's set up and ready to run. There are two scripts to launch *OpenHAB* in the main *OpenHAB* directory: **start.sh** and **start_debug.sh**. As you've probably guessed, the latter of these provides much more output about what's happening in order to help you solve any problems. To start your *OpenHAB* server, open a terminal in the place you unzipped *OpenHAB* and run:

```
./start_debug.sh
```

On a Raspberry Pi version 2 it can take a minute or so to start. Once it has, you'll see a line that ends:

```
 - Started Classic UI at /openhab.app
```

Browse to **http://localhost:8080** to see your sitemap in action. You can view this from other machines on the network by using the IP address of the hub. On our network, this is **http://192.168.0.25:8080**, but this will be different on yours. Use the static IP you set up, or enter **ip addr** at the command line to find the IP address.

Saving data

The setup we've just created will keep displaying fresh weather data every ten minutes it, but it doesn't save anything except the most recent data. In order to keep hold of historical data, you need to set up persistence. There are a few ways of doing this, but we prefer to use *MySQL*. Obviously the first step here is to install the **mysql** software:

```
sudo apt-get install mysql-server mysql-client
```

During the installation, you'll be asked to set up a root password for the database. Remember this, as you'll need it to set up the database for *OpenHAB*. Once the software is installed, you can log in to the database from a terminal with:

```
mysql -u root -p
```

Then you'll be prompted for the password you just created. To set up the *OpenHAB* database, enter:

```
CREATE DATABASE openhab;
CREATE USER 'openhab'@'localhost' IDENTIFIED BY
'openhab';
GRANT ALL PRIVILEGES ON openhab.* TO
'openhab'@'localhost';
exit;
```

Open **openhab.cfg**, find the **persistence** line and tell the system to use **mysql**:

```
# The name of the default persistence service to use
persistence:default=mysql
```

Then, further down the same file, you'll find the SQL setup. Enter:

```
mysql:url=jdbc:mysql://127.0.0.1/openhab
mysql:user=openhab
mysql:password=openhab
```

In the **configurations/persistence** directory, create a file that details when you'd like *OpenHAB* to store the data for each item. The following code tells *OpenHAB* to store the value of each change for every item:

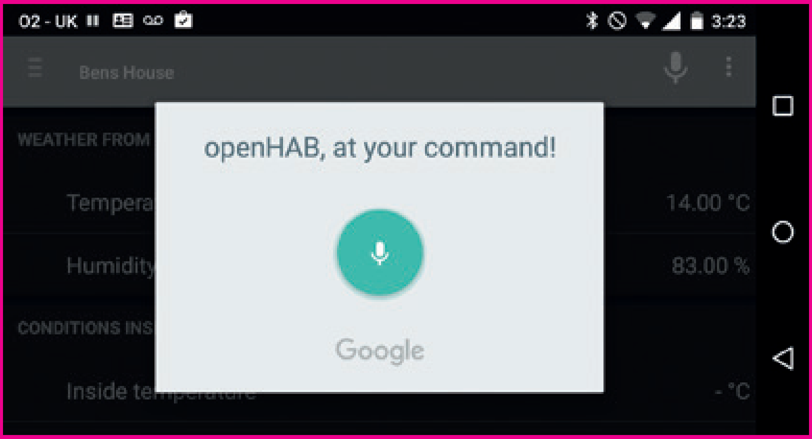
```
Strategies {
default = everyChange
}
Items {
* : strategy = everyChange
}
```

This should be called **mysql.persist**. Once these are in place, you can restart *OpenHAB* and it will pick up the new changes.

Let's do something with our stored data. *OpenHAB* includes a chart server that we can use to embed images in the site (there's also the **chart** type in the sitemap, but this doesn't work as well).

```
Frame label="Weather Chart" {
            Image url="http://192.168.0.25:8080/
chart?items=briztemp&period=D"
}
```

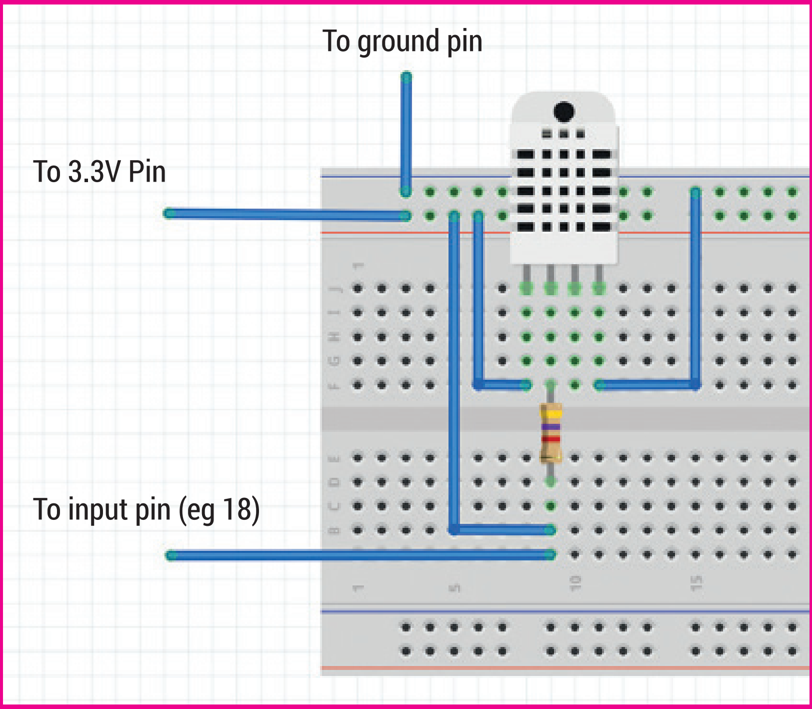
You can add more than one item and plot them against each other, as we'll see later. The period specifies the length of time that the chart should go back, and can be any one of **h,4h,8h,12h,D,3D,W,2W,M,2M,4M,Y**.



Using the OpenHAB Android app, you can control your smarthome with voice commands

ATTACHING **DEVICES**

Sensing and controlling the world.



The wiring for our temperature and humidity sensor just requires a 4.7k pull up resistor on the data pin.

Internet of Things: fad or here to stay?

When done well, the Internet of Things (IoT) gives you more control over your devices than in the past. IoT devices enable us to take computing outside the virtual world and make it interact with the real world, whether that's controlling your home, your vehicle, or anything else. We're already starting to see the benefits: cars automatically re-route based on traffic with connected navigation devices, and smarthomes adjust heating setups based on actual usage patterns of rooms. The future is connected!

Now that you've got your hub up and running, it's time to start using it to get information about your home, and use this information to make your home a better place. You can buy ready-made devices that will plug in and just work, but perhaps the most attractive thing about *OpenHAB* is that you can build your own devices to control your home in whatever way you want. Let's take a look at how this works with two addons: one that senses the environment and one that acts on it.

Our sensor will be a simple heat and humidity sensor, the AM2302 model, which can attach to any programmable controller. We'll use one of the older Raspberry Pi version 1 model B+s because we've got a few spare lying around. The first task is to install the latest version of Raspbian and connect to the network. Once you've got that, you'll need to attach the sensor. The wiring for this is all quite straightforward (see the diagram above) If you hold the Raspberry Pi with the SD card at the top and the GPIO pins on the right, the 3.3V pin is the top-left, the ground pin is the third pin down on the right, and pin 18 is the sixth pin down on the right. You'll just need a 4.7k resistor, and a way of wiring everything together (we used a breadboard and male–female headers).

Once everything's attached, it's time to set up the software side of things. Fortunately, there's a Python library for getting the information out, so it's all quite straightforward. First, you need to get the dependencies:

```
sudo apt-get install build-essential python-dev
python-openssl
```

Then you need to grab the library from GitHub and install it:

```
git clone https://github.com/adafruit/Adafruit_
Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install
```

This Python library (**Adafruit_DHT**) provides a simple read function that returns both the temperature and humidity. All we need is a simple wrapper to output either the temperature or humidity. The Python code for this is:

```
import Adafruit_DHT
import sys
pin = 18
humidity, temperature = Adafruit_DHT.read_
retry(Adafruit_DHT.AM2302, pin)
if sys.argv[1] == "temp":
        print temperature
if sys.argv[1] == "humid":
        print humidity
```

If you save this as **temphumid.py**, you can get the temperature with **python temphumid.py temp**, and the humidity with **temphumid.py humid**.

This gives us a way of getting the temperature on this machine, but we need to send it back to the hub in order for it to be useful. The easiest way to do this is through the *MQTT* messaging system. This is a really simple message-passing protocol designed specifically for the Internet of Things. The most popular Linux implementation is *Mosquitto* [sic] which comes in two parts: a client and a server. In Debian-based systems, these are in the packages **mosquitto-clients** and **mosquitto-server** respectively. All the machines we'll use *MQTT* on will need **mosquitto-clients**, while only the hub will need **mosquitto server**.

The **mosquitto-clients** package contains two tools we'll need: **mosquitto_sub**, which enables us to receive messages; and **mosquitto_pub**, which enables us to send messages. The general setup of *MQTT* is that a server maintains topics to which clients can publish messages. Once a message is published on a certain topic, it's sent to every client subscribed to that topic. In this way, it's a little like a really simplified version of IRC. Topics don't have to be predefined, so clients can subscribe or publish to any topic without any setup needed on the server.

We can use **crontab** to automatically send the temperature and humidity data every minute. Use the following command to edit the **crontab** file on the Raspberry Pi with the sensor:

```
sudo crontab -e
```

This will open a text editor with the configuration file for **cron**. You need to add the following lines:

```
* * * * * mosquitto_pub -h 192.168.0.25 -t temp -m
`python /home/pi/temphumid.py temp`
* * * * * mosquitto_pub -h 192.168.0.25 -t humid -m
`python /home/pi/temphumid.py humid`
```

The back ticks around the Python command tell *Bash* to execute this first and put the output as an argument to the **mosquitto_pub** command. On our network, 192.168.0.25 is the IP address of our hub. You'll have to change this in order for it to work for you. Exiting the text editor will automatically add these changes to the **cron** schedule.

If you followed this tutorial from the start, you'll already have copied the *MQTT* addon for OpenHAB to the appropriate directory, so the only thing left is to edit the configuration. In the **openhab.cfg** file. Find the **MQTT Transport** section and change the first configuration option to:

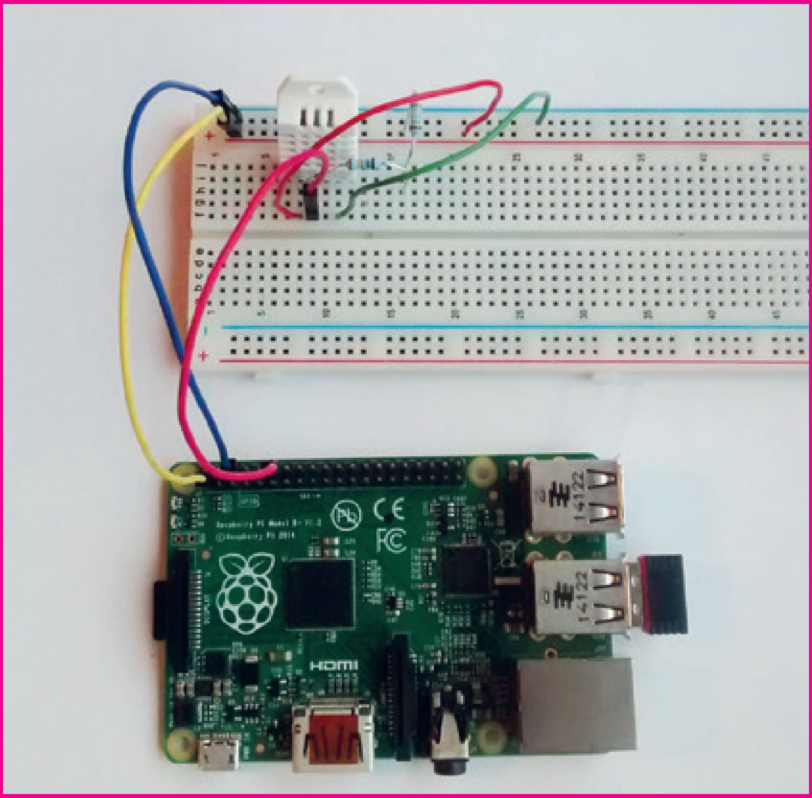
```
# URL to the MQTT broker, e.g. tcp://localhost:1883 or
ssl://localhost:8883
mqtt:msgs.url=tcp://localhost:1883
```

Now, we just need to add the configuration to the *OpenHAB* site. First, set up the following items in the **weather.items** file:

```
Number housetemp "Inside temperature [%.2f °C]" {mq
tt"<[msgs:temp:state:default]"}
Number househumid "Inside humidity [%.2f %%]"  {mqt
t"<[msgs:humid:state:default]"}
```

The data to the *MQTT* binding (in the parenthesis) first tells *OpenHAB* that this is an inbound connection (with the **<**). The second and third items are the *MQTT* service (as defined in **openhab.cfg**), and the channel. The third tells *OpenHAB* to use the state of the channel, and the final one is the filter to use (we're not using a filter so it's **default**). These items can then be entered into the sitemap.

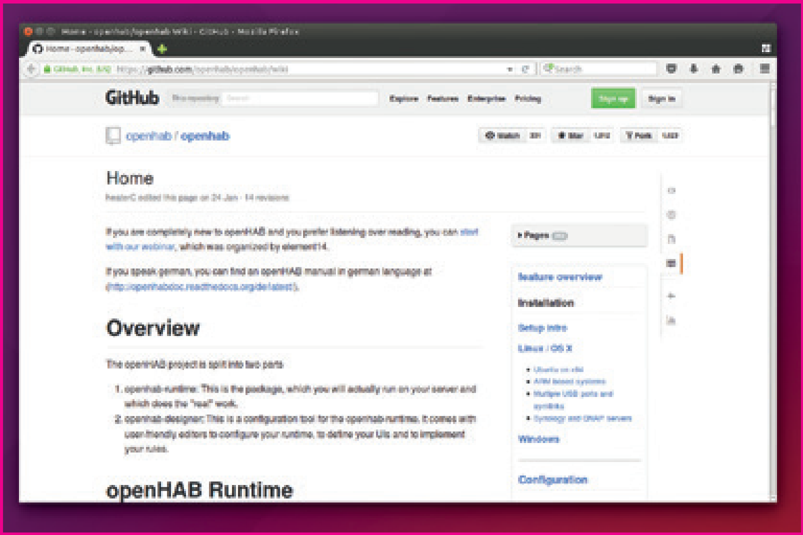
```
Frame label="Conditions inside" {
        Text item=housetemp
        Text item=househumid
}
Frame label="Weather Chart" {
        Image url="http://192.168.0.25:8080/chart?ite
ms=housetemp,briztemp&period=D"
}
```


We didn't have a 4.7k resistor to hand, so we used two 2.4k resistors in series.

This new chart will plot the temperature inside your house against the temperature outside.

Heat and humidity are great places to start, but there are loads of additional sensors you could add.

- **Passive Infra Red (PIR)** to detect movement. These are cheap and easy to connect, as they simply turn a GPIO pin on or off.
- **Bluetooth** Want to check when you leave or arrive? The chances are that your phone is publicising this information. The Bluetooth bindings enable you to monitor your whereabouts.
- **Door sensor** Check the comings and goings with a simple magnetic switch attached to your door frame.


GitHub

openhab / openhab

Home

Overview

openHAB Runtime

The *OpenHAB* wiki has full details of all the bindings, so make sure you check out compatibility before buying any hardware.

These Energenie sockets are two of the many types of you can control from OpenHAB.

- **Moisture** Ever forget to water your plants? Connect them to your *OpenHAB* hub and you can get an email (see below) whenever they run dry.
- **Smoke Alarm** As well as the usual siren, you can get a notification of any problems at home.
- **Camera** Keep an eye on what's going on in your house when you're not there.

You don't need all of these – just pick the ones that provide the data that will be useful in your smart home set up.

Get active

So far, all *OpenHAB* has done is take in information and display it on a screen. Visualising data does have some uses, but it falls far short of the idea of a smart home. Really, we should be able to control some aspect of the house. For this, you're obviously going to need some hardware that can do things in the real world. A great place to start is by experimenting with programmable sockets, which enable you to turn appliances on or off from within *OpenHAB*. You can connect them to lamps, electric heaters, coffee machines, dehumidifiers or anything else that you can control by turning on or off at the socket. As we're using a Raspberry Pi, the obvious option is the Pi-Mote from Energenie (**https://energenie4u.co.uk/catalogue/product/ENER002-2PI**), which enables us to turn sockets on or off with a simple Python command. This device can also work with devices other than the Raspberry Pi.

Again, we'll use *MQTT* to pass messages between *OpenHAB* and the process that's controlling the Pi-Mote. This means that it doesn't matter which machine runs the Pi-Mote: it could be the same machine running *OpenHAB*, or it could be some other machine. First, you need to grab the Python libraries:

```
sudo pip install paho-mqtt
git clone https://github.com/MiniGirlGeek/energenie-demo.git
```

This will create a directory called **energenie-demo**. The Python libraries aren't actually installed to the Python path, so you have to make sure that **energenie.py** from inside this directory is in the same directory as your final Python program, otherwise it won't be able to import the module properly. The easiest option is just to put your code in the **energenie-demo** folder.

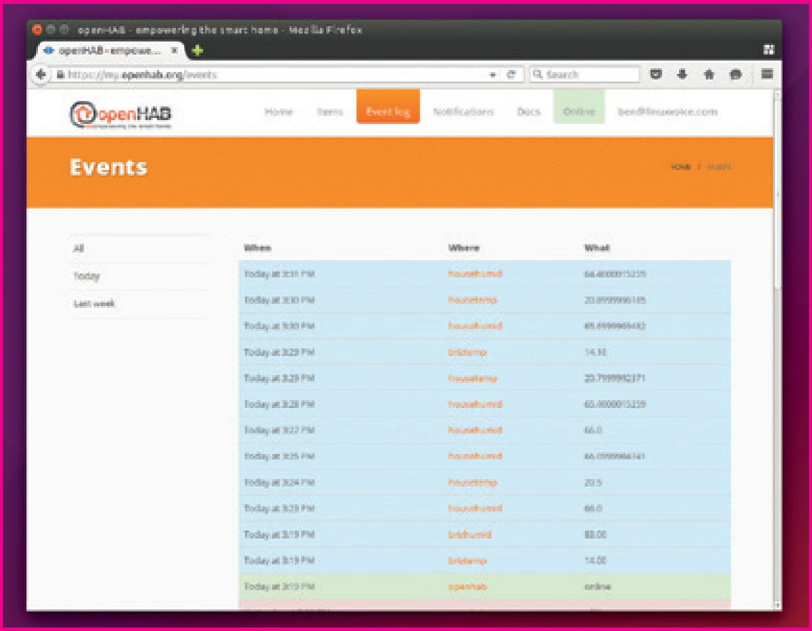
The code to turn a socket on or off on an *MQTT* command is:

```
import paho.mqtt.client as mqtt
from energenie import switch_on, switch_off

def on_connect(client, userdata, flags, rc):
client.subscribe("plugs")
def on_message(client, userdata, msg):
if str(msg.payload) == "11":
        switch_on(1)
if str(msg.payload) == "10":
        switch_off(1)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_forever()
```

This simply listens on the channel **plugs** for the message **11** (to turn on) or **10** (to turn off). We chose these because you can add additional plugs linked to the messages **21** and **20**, **31** and **30**, etc. You'll need to start this running before moving on to the *OpenHAB* setup.

Now we have these, we need to create the items and add them to the sitemap. The items use the *MQTT* bindings in a fairly similar way to the

The My.OpenHAB service enables you to access your data via a Rest API for integration with other services.



Events

When	Where	What
Today at 3:31 PM	househumid	64.4000015259
Today at 3:30 PM	housetemp	20.8999996185
Today at 3:30 PM	househumid	65.6999998482
Today at 3:29 PM	bridtemp	14.10
Today at 3:29 PM	housetemp	20.7999992371
Today at 3:28 PM	househumid	65.4000015259
Today at 3:27 PM	househumid	66.0
Today at 3:25 PM	househumid	66.0999984741
Today at 3:24 PM	housetemp	20.5
Today at 3:23 PM	househumid	66.0
Today at 3:19 PM	bridhumid	83.00
Today at 3:19 PM	bridtemp	14.00
Today at 3:19 PM	openhab	online

temperature and humidity sensor. The **item** line is:

```
Switch plug1 {mqtt=">[msgs:plugs:command:ON:11],>[
msgs:plugs:command:OFF:10]"}
```

Here you'll notice that the command starts with a 'greater than' sign, which means outputs. The first two arguments are the **mqtt** service and the channel. The third tells *OpenHAB* that it should operate on a command rather than a state; the fourth is which command to operate on; and the final one is what to send when that command is issued. There are two sets of arguments because there are two possible commands.

Item is then placed in the sitemap with the following:

```
Frame label="Sockets" {
        Switch item=plug1
}
```

If you want to add more sockets (a Pi-Mote can handle up to four), you just need to add additional commands to the Python file, and associated items and sitemap entries.

Switching a socket on or off is one of the simplest things that a smart home can do, but there's far more if you want to get the hardware:

- **Radiator and heating control** This is one of the biggest advantages of smart homes, and there are several options that will enable you to completely program your heating setup.
- **Light switches** Commercially available light switches can be used both via *OpenHAB* and as regular switches.
- **Light bulbs** Control the colour and brightness of your lights from your computer.

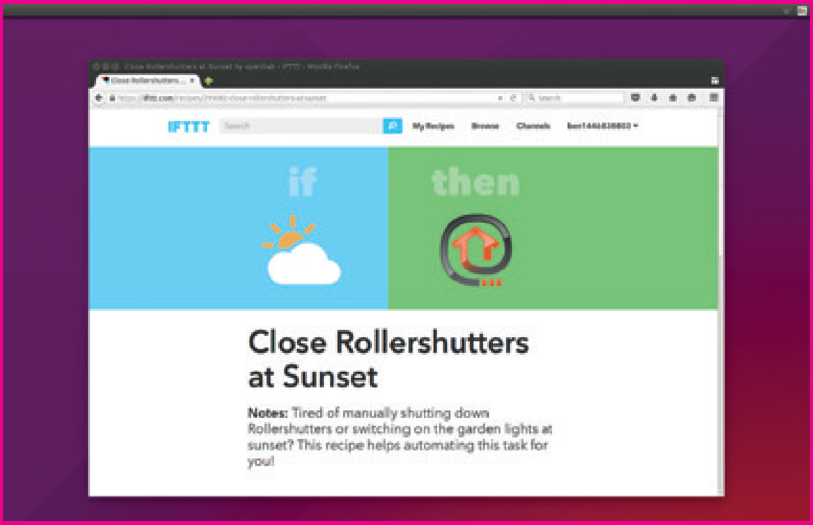
Obey!

Adding devices that can interact with the real world makes our *OpenHAB* setup useful, but it doesn't make it smart. For that, we need to add some basic intelligence. This is done with rules that go in the **configurations/rules** folder and have the **.rules** extension. The format is:

```
rule "rule name"
when
        <condition>
then
        <action>
```

The code for **condition** and **action** are written in a specialised version of Java. Here's a simple rule to turn on a socket when the temperature gets cold, and turn it off again if it gets hot:

```
rule "temp control"

when
        Item housetemp changed
then
        if(housetemp.state < 19) {
                sendCommand(plug1, ON)
        }
        if(housetemp.state > 22) {
                sendCommand(plug1, OFF)
        }
```


Close Rollershutters at Sunset

Notes: Tired of manually shutting down Rollershutters or switching on the garden lights at sunset? This recipe helps automating this task for you!

If This Then That (**https://ifttt.com**) contains lots of premade rules that you can take inspiration from, many of which work with *OpenHAB*.

Open access

The *OpenHAB* setup we've created here provides you insight and control over your home, but only while you're connected to your local LAN. If you want to access it from outside your home you'll need to setup external access. There are a couple of ways to do this. There's a service called My OpenHAB at **my.openhab.org** that connects your home setup with a web server that both stores your data and enables you to use it in external services, and provides you access to your *OpenHAB* interface from where ever you are on the internet.

One of the reasons we gave for using *OpenHAB* at the start was because we didn't want to hand all our data over to a cloud provider, yet My OpenHAB goes against this. If you want to keep complete control over your data, the alternative option is to configure your internet router to forward all connections on port 8080 to port 8080 on your *OpenHAB* server. You'll then need to sign up to a dynamic DNS service that will link your home IP address with a domain name and allow you to access it when you're on the go even if your ISP changes your IP address.

Each home is different, so there's no right way to set up a smart home. The best aspect of *OpenHAB* is that it doesn't try to dictate a way you should work – instead it gives you the tools to work with whatever hardware you want to get in whichever way you want to work. This does mean that it's more work to set up than almost any commercial offering, but at the same time, it makes it far more valuable once it is set up.

My OpenHAB and If This Then That

OpenHAB rules are really powerful and can interact with lots of external services including email and Twitter (provided you have everything set up in **openhab.cfg**). An alternative option is linking a My OpenHAB account to If This Then That (IFTTT). IFTTT is an internet service for making actions based on inputs. It's far simpler that the OpenHAB rules, but it's also easier to use and can interact with more external services.

Using IFTTT, you can set rules based on anything from events from Google Calendar or Facebook to new items on shopping sites such as Ebay to location data from Android devices. The range of possibilities is really staggering. Sign up at **http://ifttt.com**.

SUSEcon 2015

Mike Saunders checks out the new OpenSUSE release, big-iron hardware from IBM, and toy penguins from… Microsoft?!

If you're on the hunt for a Linux-related job, SUSE may be worth checking out. At the recent SUSECon 2015 conference, President and General Manager Nils Brauckmann was chuffed to announce that 20% of the company's current employees were taken on in the last year. And at the time of writing, 67 positions were open, so visit **www.suse.com/company/careers** to see what's on offer.

But what about the conference itself? Linux Voice was there of course – and thanks to the competition we ran back in issue 20, three of our readers got to attend for free as well. Held in the Beurs van Berlage, a former commodities exchange in central Amsterdam, SUSECon 2015 was the biggest event yet in SUSE's history with around 1,000 attendees. And, of course, many engineers, testers and product managers from SUSE were present. Much of the event was geared towards SUSE partners and users in large enterprises, but there was plenty for us hobbyists and general Linux geeks to explore as well.

IBM, for instance, showed us that the mainframe is far from dead with its LinuxONE range of machines. These mighty boxes – based on the z13 mainframes – are designed to handle jobs involving huge numbers of transactions per second (such as share trading). IBM claims the LinuxONE Emperor machine, the most powerful on offer, can run 8,000 virtual machines simultaneously (and thousands of containers inside each one) without breaking a sweat. The company has uploaded a video demonstrating a LinuxONE handling vast workloads at **https://www.youtube.com/watch?v=VWBNoIwGEjo**.

Mainframes have historically been enormously expensive beasts, but with the LinuxONE IBM is trying to attract users on tighter budgets as well. So there's a cut-down model called the Rockhopper, which sports

Linux Voice travel protip: Gandhi's restaurant in the centre of Amsterdam does awesome Indian food.

IBM's LinuxONE mainframes can chew huge amounts of data in a single box – here's one in a "naked" form.

Microsoft has come a long way since the days of calling Linux a "cancer"

an "elastic pricing" model – ie you install the machine on site in your company, and then only pay for the computing resources you use.

Breakout sessions at the event were used to show what's new in upcoming SUSE Linux Enterprise service packs, and demonstrate various hot technologies like Docker in action. On the show floor, SUSE partners and other Linux-related businesses were touting their wares – and one company caught us by surprise. A certain company that once called Linux a "cancer". Yes, Microsoft was there, even giving away plushy penguin toys to promote the company's open source efforts. We posted a photo on Twitter and our followers were quick to respond, warning us that it could be a Trojan horse, or we should put it in a Faraday cage to protect us from its mind-altering waves. Pro-open source moves from Microsoft are welcome, but it's healthy to stay cynical...

One giant leap

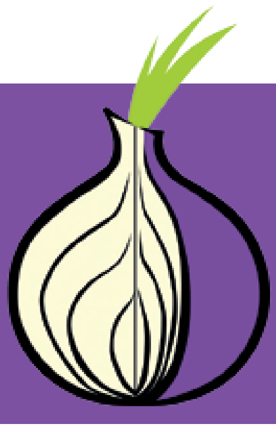
The biggest news for us, though, was the release of OpenSUSE 42.1 "Leap". Given that the previous release of the distro was 13.2, you might be wondering about the significance of this big version number bump. Well, Richard Brown of SUSE told us that the ideas behind this release started off under the internal SUSE codename of "Project 42" – and that number has stuck ever since. But what makes it so different to the previous versions?

Well, this is the first OpenSUSE to be based on the SUSE Linux Enterprise sources. SUSE wants to consolidate the efforts across its two distros – but not to end up with two functionally identical products like Red Hat has with RHEL and CentOS. No, Brown and his team wanted to do "something more interesting" with the opening of the SUSE Enterprises sources. So OpenSUSE 42.1 "Leap" takes the base packages from the enterprise release – the kernel, libraries and similar low-level plumbing – and the community can add newer packages where appropriate for home users, small businesses and desktops (eg newer versions of *Firefox* or *LibreOffice*).

So the end result is a community-supported distro with a reliable base, but moving at a slightly faster pace than the more conservative enterprise options. OpenSUSE Tumbleweed – the rolling release distro – will continue, and Richard Brown even claimed that rolling releases are the future of all distros, including enterprise-oriented ones. This might seem crazy when big businesses rely on stability above all else, but Brown is confident that with automated testing (like SUSE's OpenQA), and atomic updates (making it easy to roll back a package if something breaks), a rolling release geared towards enterprise users may be possible one day down the line. Watch this space!

Technology showcases provided glimpses of apps and hardware being built around SUSE's distros.

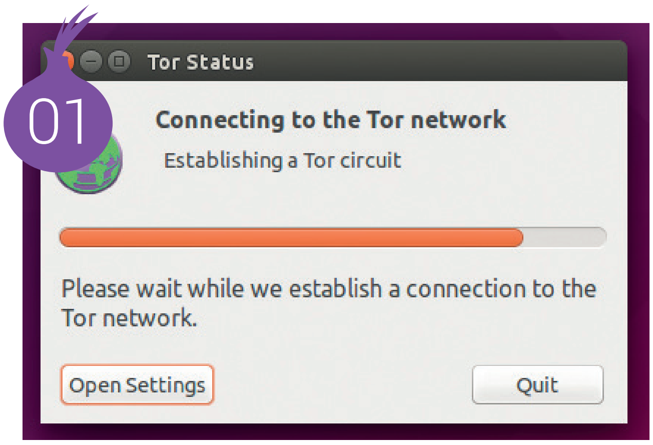
SECRETS OF THE TOR BROWSER



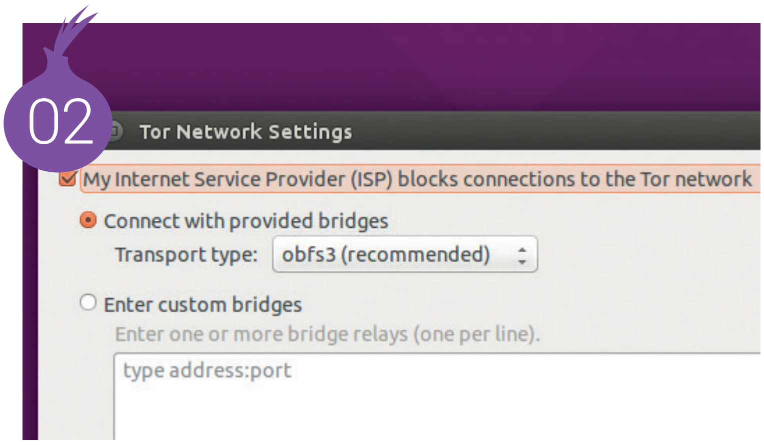
The ultimate tool for self-defence when browsing the web.

You are being watched. Every click, keystroke and command you enter into your web browser is recorded by advertisers and governments. In order to have any hope of online privacy, you need to change the way you go online. Only by hiding your IP address and location can you stop this routine invasion of your personal data.

The *Tor Browser* is a simple defence tool. You just need to download and extract the software, then run **start-tor-browser** to launch a protected web session. The software will automatically connect to the *Tor* network, which uses three encrypted network hops to separate your browsing activity from any identifying information. Since the people most in need of protection online aren't necessarily tech-savvy, the *Tor Browser* is designed to be easy to use, and it succeeds wonderfully at this. There are, however, a few additional features available to help you customise your browsing experience if you peer beneath the slick exterior.



Connecting to the Tor network
Establishing a Tor circuit
Please wait while we establish a connection to the Tor network.



☑ My Internet Service Provider (ISP) blocks connections to the Tor network
● Connect with provided bridges
Transport type: obfs3 (recommended)
○ Enter custom bridges
Enter one or more bridge relays (one per line).
type address:port

01 Connect directly

In normal usage, you just need to start the *Tor Browser* and the software will automatically connect to the *Tor* network before opening the browser window.

The first page you'll see will be a *Tor* test page that checks that everything has worked (if the page turns red, something's gone wrong, and you're not browsing anonymously). Once this is done, you'll be able to browse the web in anonymity.

02 Stealth connect

If your ISP or government blocks direct access to the *Tor* network, you can get in via a bridge. The recommended option is *obfs3*, which scrambles the communication in an attempt to disguise the type of traffic as it connects to a private entry point into the *Tor* network. Other options include *Flash proxy*, which uses browser-based bridges, and *Meek*, which hides the *Tor* data in connections to popular internet services.

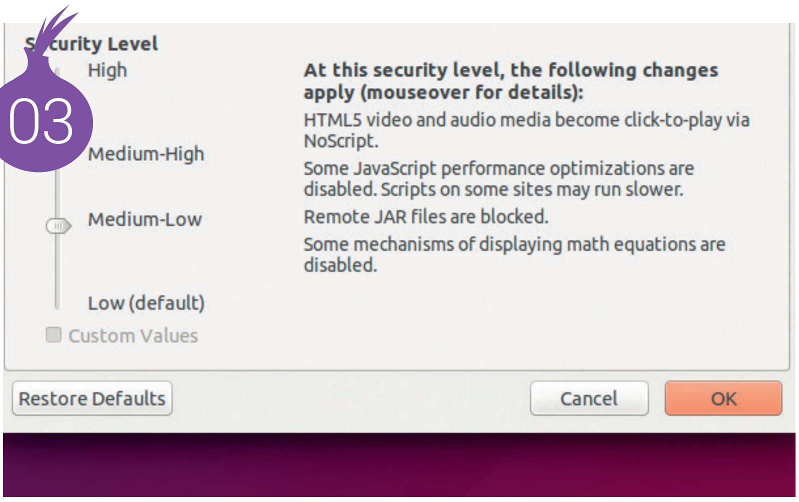
03 Privacy slider

There are some settings that can increase your privacy, but make some websites hard to use – for example, limiting the use of font features such as different sizes and typefaces. You can choose your own balance between browser usefulness and privacy with *Tor* privacy settings, which enable you to make the right choice between privacy and convenience. The defaults will be sufficient for most people, but if you're facing a serious, persistent attack then you may need to move the slider up to keep your connection private.

The HTTPS Everywhere plugin forces your browser to use the encrypted version of a website (via HTTPS) whenever there's one available, rather than the unencrypted version

04 HTTPS Everywhere

The *HTTPS Everywhere* plugin created by the Electronic Frontier Foundation forces



Security Level
High
Medium-High
Medium-Low
Low (default)
Custom Values

At this security level, the following changes apply (mouseover for details):

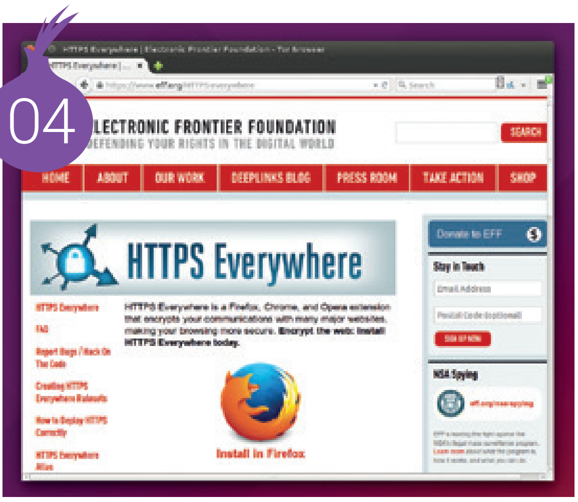
HTML5 video and audio media become click-to-play via NoScript.

Some JavaScript performance optimizations are disabled. Scripts on some sites may run slower.

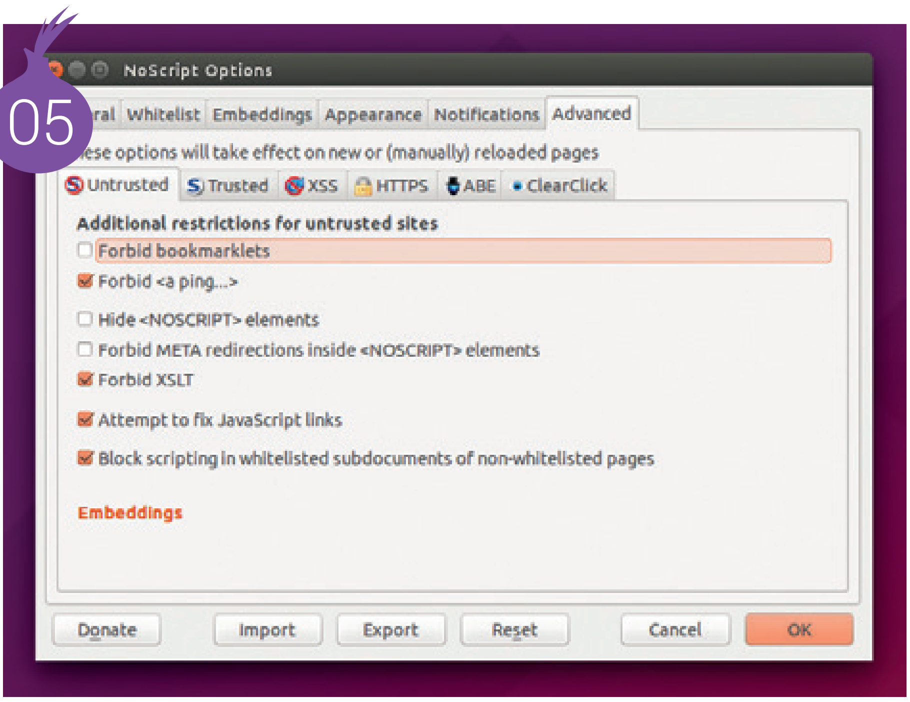
Remote JAR files are blocked.

Some mechanisms of displaying math equations are disabled.

Restore Defaults · Cancel · OK



HTTPS Everywhere is a Firefox, Chrome, and Opera extension that encrypts your communications with many major websites, making your browsing more secure. Encrypt the web: Install HTTPS Everywhere today.



ral · Whitelist · Embeddings · Appearance · Notifications · Advanced

ese options will take effect on new or (manually) reloaded pages

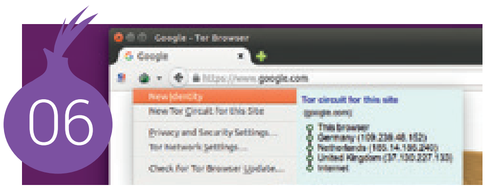
Untrusted · Trusted · XSS · HTTPS · ABE · ClearClick

Additional restrictions for untrusted sites

- ☐ Forbid bookmarklets
- ☑ Forbid <a ping...>
- ☐ Hide <NOSCRIPT> elements
- ☐ Forbid META redirections inside <NOSCRIPT> elements
- ☑ Forbid XSLT
- ☑ Attempt to fix JavaScript links
- ☑ Block scripting in whitelisted subdocuments of non-whitelisted pages

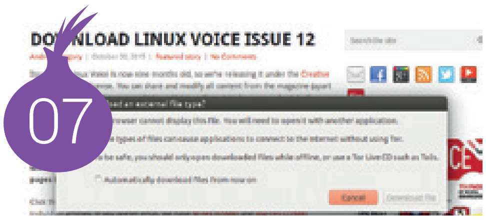
Embeddings

Donate · Import · Export · Reset · Cancel · OK



New Identity
New Tor Circuit for this Site
Privacy and Security Settings...
Tor Network Settings...
Check for Tor Browser Update...

Tor circuit for this site (google.com)
This browser
Germany (109.238.48.152)
Netherlands (185.14.185.240)
United Kingdom (37.130.227.133)
Internet



oad an external file type?

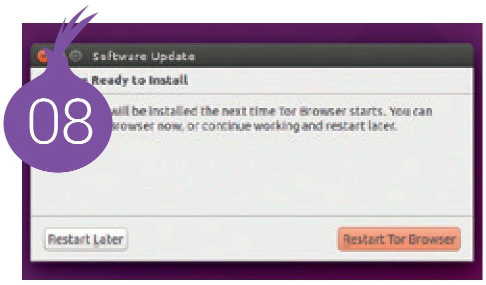
rowser cannot display this file. You will need to open it with another application.

types of files can cause applications to connect to the internet without using Tor.

be safe, you should only open downloaded files while offline, or use a Tor Live CD such as Tails.

☐ Automatically download files from now on

Cancel · Download file



Ready to Install

will be installed the next time Tor Browser starts. You can rowser now, or continue working and restart later.

Restart Later · Restart Tor Browser

your browser to use the encrypted version of a website (via HTTPS) rather than the unencrypted version (via HTTP) whenever both are available. This is important because even though *Tor* anonymises web browsing, it can't encrypt the final transmission from the *Tor* network to the web server. Using HTTPS means your browsing data is never sent unencrypted.

05 NoScript

Although most scripts that run on websites are perfectly harmless, there are a few that can be used to track you or attack your browser. To help defend against this threat, the *Tor Browser Bundle* comes with *NoScript*, a plugin that enables you to control what scripts run on your machine and which websites run them. You can change the settings, but be careful as this can lead to leaking data.

06 New identity

When connecting to the web via *Tor*, your traffic is routed through an exit node. This can be used to track you across a site (the site can't tell who you are, but they can see that a browser has been through the pages you've been through). The *Tor Browser* gives you the ability to change to a new exit node, thereby getting a new identity and breaking the tracking that the site is using.

07 Warnings

Tor can't protect everything you do online, so it comes with warnings for when you try to perform an action that could compromise your anonymity (such as downloading a file that could contain links to external resources or malicious code). It's then up to you whether or not you heed these warnings – clever as it is, software can't do everything for you!

08 Update check

It's important to keep all your software up to date, but with security-critical software it's especially important, and the *Tor Browser* is one such program. It's also especially prone to slipping out of date because it's usually installed outside your distro's package manager. The update checker is a really simple tool for making sure that you're always protected by the latest security updates, so it's a good idea to run it every time you start the browser.

Inside CoreOS

Dive with us into the world of complicated cluster computing with **Graham Morrison** and **Brandon Philips**, co-founder of one of the hottest Linux startups in the galaxy.

There are many outcomes that Linus Torvalds could never have envisaged when he embarked upon his quest to create the Linux kernel. One is that it would lead to the creation of an operating system that now dominates the computing hemisphere. Another would be that by choosing to license the kernel under the terms of GPLv2 he would change an industry's attitude towards open source and Free Software. But for us, two of the most unpredictable outcomes are how Linux has become used at the small and large scale. At the small scale, Linux-based Android is the dominant platform, fuelling the smartphone revolution. Those smartphones have spawned a new era and expectation for connectivity and data-driven intelligence. This has fuelled the other outcome, Linux at a large scale, in the cloud.

CoreOS Linux is a minimal Linux distribution that predates and encompasses the recent and most definitive stages of the cloud revolution – the migration from discrete distribution installations to what are now being called clusters of containers. It even boasts kernel supremo, Greg Kroah-Hartman, as one of its chief advisors, and another kernel ace, Matthew Garrett, is an engineer at CoreOS.

This makes CoreOS a wonderful project to study if you want to understand how and why the cloud evolved from those old servers and infrastructure into this buzzword-infested computing paradigm. Which is why, when had the chance to spend some time with Brandon Philips – CTO and co-founder of CoreOS, and one of its original developers – we couldn't pass up on the opportunity to ask about Linux, hype and containers.

CoreOS is a minimal Linux distribution that encompasses the recent and most definitive stages of the cloud revolution

"My co-founder, Alex Polvi, and I had been in infrastructure software for a while," Brandon started out by telling us, and it was clear from the beginning that Alex and Brandon had no intention that their new distribution would be encumbered with the same problems that held other distributions back, especially when it came to the dynamism and portability demanded by new online businesses.

"I worked at SUSE Linux... on the Linux kernel and putting together a distro and fell out of the world of enterprise distros. [Alex and I] had known each other for around 12 years. I was looking for a change – and he was kind of just floating around, so we got together and said, 'How can we fundamentally fix and secure how this back-end infrastructure stuff works.' We tossed around ideas and CoreOS Linux was the thing we settled on."

Come in from the cold

Cloud has been much derided over the last decade, most famously by Richard Stallman, who told the *Guardian* in 2008 that the term is "worse than stupidity: it's a marketing hype campaign."

This is true of 2008, but by the time the first lines of CoreOS were being committed in March 2013, we think the term had started to have concrete meaning of its own, describing a solution to a very specific set of problems. Brandon describes these problems when he told us about how CoreOS got started.

"There's a lot of different places we could have attacked, but it goes all the way back to the distro," he told us. "The way we're putting distros together today makes it very difficult to have automatic updates, like we have on our phones – Android and iPhones, and it makes it very difficult because this API boundary is so huge that the distro has to maintain. Your database comes from your distro. Your language runtime comes from your distro. Your kernel comes from your distro. And it's this huge sprawling collection of APIs and it makes it impossible so say, 'Right, we're going to move from version A to version B of our Linux distro,' and have confidence in that actually working. Because suddenly you have a new database version, you have a new language version, it's like this huge zero day switch. I saw it at SUSE. There were people running versions of SUSE that were 12 years old, because it's critical infrastructure and because their application is tightly bound with this huge API layer. There was no way for them to move from A to B."

Enterprise: warp speed

The idea of the 'old enterprise' came up several times during our conversation. It was also at the heart of a point made by Greg Kroah-Hartman in his late-2014 'Ask Me Anything' on Reddit when asked specifically about CoreOS. "I'm really happy with CoreOS, it's the way I think 'enterprise' distros should be developed and deployed," he wrote, before explaining, " Lots of things have changed over the years and trying to keep servers as 'pets' is not the way to do it. Large numbers of servers, deploying services that you need to use/provide is the way to go. Keeping those servers up to date is hard, as is managing the tasks on those servers, and CoreOS provides a way to help out with that."

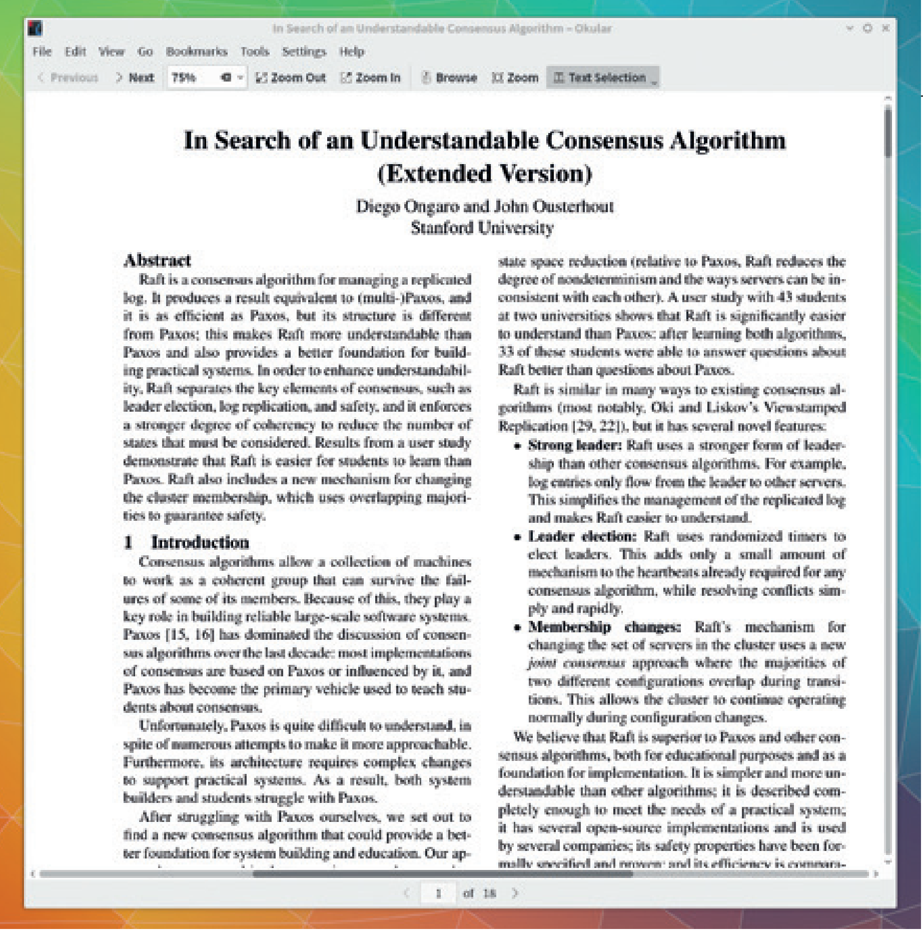
What started to make the difference was the change in how containers were used, from virtual machines to what have become isolated sandboxes. CoreOS, in particular, was one of the first projects to use containers to address a Linux-specific problem – transparent and secure package management that could be easily copied across a huge cluster of machines. Docker, the hugely successful deployment engine that now dominates the cloud, has since helped to solve this problem, but as Brandon explained, this was a time before Docker.

"Containers are necessary for CoreOS Linux to work and so we we're serving that scene and we were starting to build some stuff, an API-driven container system [the prototype is still up on GitHub]. Around that time, about 2–3 months after we'd got the distro and good firmware and were able to generate builds and document it, Docker came out. We'd known Soloman [Hykes, creator of Docker] and the team, and it totally made sense to band on what we were working on and join up."

Brandon was contributing to Docker, even joining the project's governance board, while at the same time trying to figure out how to solve the combined problem of package updates and security rollouts. The answer was always going to be containers.

"We were thinking through the problems and possible solutions, and the obvious solution was for containers," he said. "The big risk we thought was that, we'll build this CoreOS Linux thing, but Linux containers have existed for eight years or more and nobody was using them in the way we were thinking about. Instead of it being a lightweight virtual

"We've got a little over 20 engineers, and that continues to grow. We've got offices in San Francisco and New York, and we're always looking to hire" Brandon Philips.


In Search of an Understandable Consensus Algorithm – Okular
File Edit View Go Bookmarks Tools Settings Help
Previous
Next
75%
Zoom Out
Zoom In
Browse
Zoom
Text Selection
1 of 18

In Search of an Understandable Consensus Algorithm (Extended Version)

Diego Ongaro and John Ousterhout
Stanford University

Abstract

Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems. In order to enhance understandability, Raft separates the key elements of consensus, such as leader election, log replication, and safety, and it enforces a stronger degree of coherency to reduce the number of states that must be considered. Results from a user study demonstrate that Raft is easier for students to learn than Paxos. Raft also includes a new mechanism for changing the cluster membership, which uses overlapping majorities to guarantee safety.

1 Introduction

Consensus algorithms allow a collection of machines to work as a coherent group that can survive the failures of some of its members. Because of this, they play a key role in building reliable large-scale software systems. Paxos [15, 16] has dominated the discussion of consensus algorithms over the last decade: most implementations of consensus are based on Paxos or influenced by it, and Paxos has become the primary vehicle used to teach students about consensus.

Unfortunately, Paxos is quite difficult to understand, in spite of numerous attempts to make it more approachable. Furthermore, its architecture requires complex changes to support practical systems. As a result, both system builders and students struggle with Paxos.

After struggling with Paxos ourselves, we set out to find a new consensus algorithm that could provide a better foundation for system building and education. Our ap-

state space reduction (relative to Paxos, Raft reduces the degree of nondeterminism and the ways servers can be inconsistent with each other). A user study with 43 students at two universities shows that Raft is significantly easier to understand than Paxos: after learning both algorithms, 33 of these students were able to answer questions about Raft better than questions about Paxos.

Raft is similar in many ways to existing consensus algorithms (most notably, Oki and Liskov's Viewstamped Replication [29, 22]), but it has several novel features:

- **Strong leader:** Raft uses a stronger form of leadership than other consensus algorithms. For example, log entries only flow from the leader to other servers. This simplifies the management of the replicated log and makes Raft easier to understand.
- **Leader election:** Raft uses randomized timers to elect leaders. This adds only a small amount of mechanism to the heartbeats already required for any consensus algorithm, while resolving conflicts simply and rapidly.
- **Membership changes:** Raft's mechanism for changing the set of servers in the cluster uses a new *joint consensus* approach where the majorities of two different configurations overlap during transitions. This allows the cluster to continue operating normally during configuration changes.

We believe that Raft is superior to Paxos and other consensus algorithms, both for educational purposes and as a foundation for implementation. It is simpler and more understandable than other algorithms; it is described completely enough to meet the needs of a practical system; it has several open-source implementations and is used by several companies; its safety properties have been formally specified and proven; and its efficiency is compara-

The white paper on the Raft consensus algorithm describes a method that enables a cluster of servers to make decisions based on single database shared across the cluster.

machine, we're thinking of it as, 'What a replacement for the package manager!' It's a different way of packing the software. A different way of doing a distro."

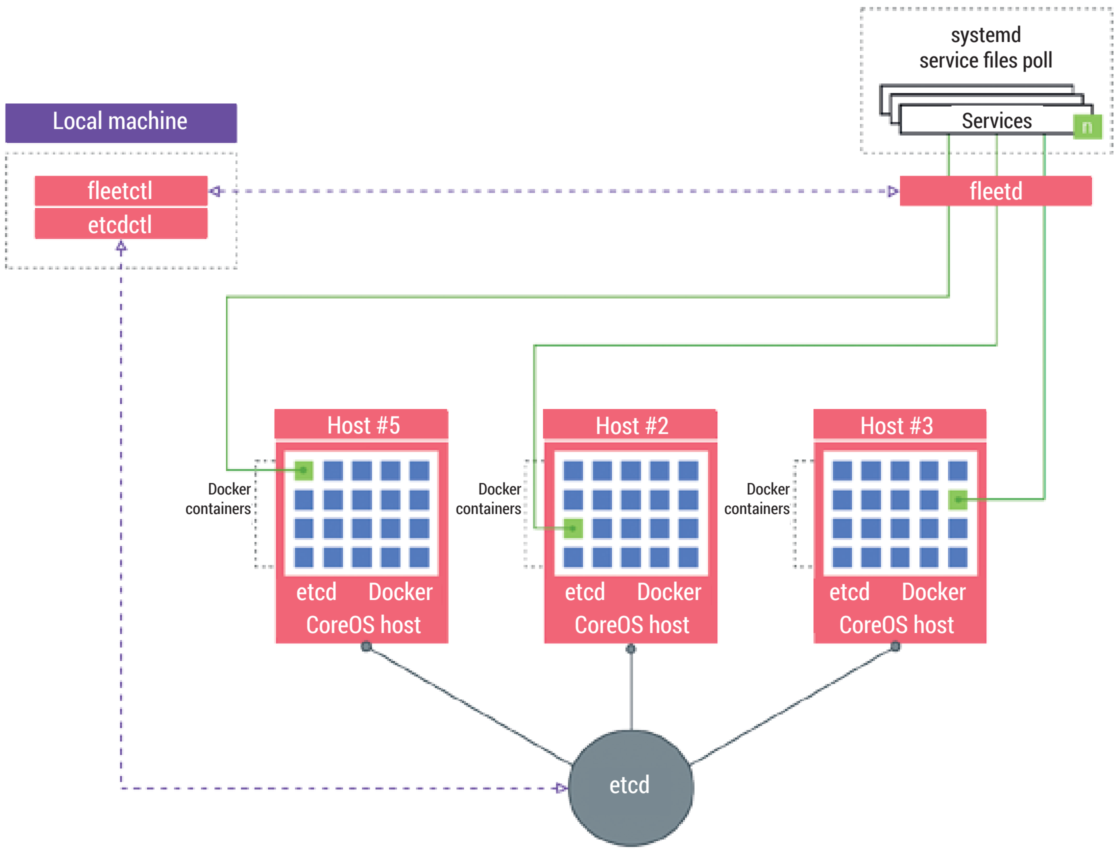
Generation next

This is where there's a demarcation between what was the old cloud – the cloud Richard Stallman was talking about – and what has become a new kind of super-networked pervasive infrastructure.

Cloud initially seemed to be a re-branding of ideas that have been around for decades, like server storage, thin clients and distributed computing. But the ideas behind cloud computing have started to coalesce thanks to the huge data being generated and demanded by the hundreds of internet enabled devices that we come in contact with. Google, Amazon, Facebook, Twitter and the thousands of smaller companies whose businesses are thriving online owe their dynamism to many instances of Linux running in sympathy with one another. Cloud now means being able to scale, only paying for resources you use, and abstracting the applications and services that run in the cloud out of the hardware.

After deciding on containers, the next challenge for CoreOS was to create a centralised configuration system. This is a much more complex problem than it sounds, especially when dealing with clusters of potentially hundreds of environments. There needs to be persistent storage, even when 'nodes' in the network go down, while also providing access to things like service registration, service discovery and process coordination across the cluster. CoreOS came-up with 'etcD', a clever, open source persistent data store that's now a vital component in Google's Kubernetes cluster manager.

"We built a lot of difficult technology with the consistent key-value store called etcD – the underpinnings of Kubernetes, which is essentially containers. etcD is the way all the important information and the consistency of the clusters is maintained. This was necessary because we had this vision of automatically updated Linux server hosts. But we also didn't want to tell the other infrastructure



CoreOS is a minimal Linux distribution designed to be run in parallel with each host running multiple containers running their own applications.

people, 'Yeah, we're going to automatically update and you won't have any control or idea of when that's going to happen.' You need this distributed database so that a machine can acquire a lock and say 'I'm rebooting for an update'."

This is technical stuff – the traditional domain of computer science. Like the hive mind, the etcD database lives on multiple machines. We wanted to know how the system was maintained when it didn't exist on a single machine, when there's no single process running to make a single decision based on the data. Brandon explained that this decision-making was made by a 'consensus algorithm' implemented using something called the 'Raft Protocol'.

"We got really lucky that there's a PhD student from Stanford University called Diego Ongaro who had just written a very well designed consensus algorithm in a white paper called Raft. We were one of the first implementations of it," Brandon told us, before explaining how they implemented Raft at CoreOS.

"The lead engineer on etcD, Xiang Li, was a summer intern and we told him, 'We need a consistent data store.' and he went off and built it. He is now mentioned as one of the authors on the thesis of Raft. So we were very early in the formative time of that algorithm and you can see it used all over the place now, and our implementation is one of the best because we've been there from the beginning."

Kubernetes is an integral part of the Cloud Native Computing Foundation, which was launched at the same time by the Linux Foundation. CoreOS has advocated for lots of design decisions as well as adding authentication back-ends such as OAuth. Kubernetes is important because it represents the next generation of cloud infrastructure, the kind of infrastructure that Google has developed internally.

Finally, we wanted to know whether Brandon thought CoreOS had found its fight and whether the journey had been worth it.

"We started two and a half years ago with this vision of how we'd like infrastructure to happen. Now we have Google and we have all these people who are agreeing and we're bringing open source software behind that vision. So like Cloud Native Computing Foundation, it's an amazing change from 2.5 years ago – here are some people hacking in some garage in Palo Alto, kind of being called crazy people, to now an entire industry groundswell of new technologies in open source happening. It's been fun."

OPEN SOURCE OR FREE SOFTWARE

On the emerging fault line between permissive and copyleft licences.

What's your commitment to open source?

Brandon Philips: People are unwilling to bet their infrastructure choices on software that isn't open source. And we're willing to go in and fill in the white space when things are missing. We built things like etcD, key value store; we built things like Flannel, which is a networking system, we've contributed to other projects that we didn't originate, like Kubernetes and Docker and the Linux kernel and the Grub bootloader and Systemd.

Would you recommend permissive licences to your customers?

BP: [TheGPL] scares people off and people who aren't going to contribute are always going to find a work around, even of the GPL. They'll find some way – they'll just open source the patches and code-drop them on some FTP site when they release the product and they'll never get merged upstream, or something. And in the case of the Linux kernel, it actually is a more reasonable licence, because, say I get a wizbang widget from Amazon – it's a Wi-Fi router or something, [the licence] is reasonable, because if that provider ever goes away I want to secure that thing, I want to have the device drivers necessary to patch the kernel. In the case of a lot of this server software, like etcD is under the Apache licence, you're not going to make modifications that are specific to a piece of hardware and if you're going to contribute you're going to contribute.

Are permissive licences seen as the path of least resistance?

BP: Yeah, and it doesn't scare people so much either. I think a lot of credit needs to be given to the Free Software movement in general because in a lot of cases infrastructure people simply demand that they have access to the source code. It does go all the way back through the history of UNIX – if you're running a big complicated system and something goes wrong you really appreciate the ability to look at the source to see what the computer was told to do versus what the computer is doing right now.

FAQ

Flutter

Oh look, yet another mobile app development framework! Except this one has a few aces up its sleeve...

MIKE SAUNDERS

Q Do we really need another framework for developing mobile applications? Can't the world just standardise around one?

A Well, to quote Minix creator Andrew Tanenbaum: "The nice thing about standards is that you have so many to choose from". It's true that there are new mobile frameworks popping up every day, and the vast majority of them have their five minutes of fame on Hacker News before disappearing into the void forever. Most of them promise the world – extremely rapid development of feature-rich apps that run seamlessly across Android, iOS and other platforms – while bringing along their own sets of problems (such as codebase immaturity and lack of commercial or community support).

But before we get on to Flutter, let's clarify what an app development framework actually is. Essentially, a framework is a collection of components designed for building software, including compilers, libraries and toolkits. Imagine you want to make an Android application for an online shop, for instance: you'd need a compiler to turn your human-readable source code into bytecode that can be processed by the Android Runtime on the device. You'd also need a bunch of libraries to provide, encryption, online communication and other features (assuming you didn't want to write them all yourself). Finally, you'd need a toolkit for creating forms, buttons and other interactive widgets on the screen.

Q Doesn't the Android SDK provide all that already?

A Yes, but – of course – it's limited to making apps for Android. Flutter lets you create software that runs on both Android and iOS, so you can write programs that run on the vast majority of mobile devices from a single code base. On top of that, these programs will look good. Flutter's own description says that the framework "gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps on both Android and iOS. Flutter gives users beautiful, fast, and jitter-free app experiences."

Q So, more big promises. What makes this one different?

A Well, it's open source for starters. But most significantly, it's backed by Google. You might think that's no huge reason to celebrate; after all, Google has hyped up many of its own products, services and projects over the years, only to abandon them down the line (like Buzz and Wave). So Google's support doesn't guarantee that Flutter is 100% the future of mobile application development, but it can reassure us that it's not a fly-by-night project that will be dropped when the developers find new toys to play with.

Some mobile app development frameworks have made extensive use of JavaScript and HTML 5, the idea being that many people already know these languages, so it should be fairly easy to knock together software that can be run in a desktop browser or wrapped up into an app on Android or iOS. All you need is a HTML rendering engine – included as standard in mobile devices – and you're good to go.

Flutter rejects this approach, however, for performance reasons. "HTML and WebViews as they exist today make it challenging to consistently hit high frame rates and deliver high-fidelity experiences, due to automatic behaviour (scrolling, layout) and legacy support." So if you want to write an app that's extremely responsive and fluid, HTML and JavaScript simply won't cut it.

Q So what does Flutter use instead?

A Dart. This is a fairly new open source language developed at

With Flutter you can write programs that run on the vast majority of mobile devices – Android and iOS – from a single code base

Google that was first unveiled to the world in late 2011. The current stable release at the time of writing is 1.12, and while Dart is still a baby compared with many other big-name programming languages like C++ and Java, it is already being used extensively inside Google.

Dart is an object-oriented language with a C-like syntax and support for optional typing; you can run Dart programs in "checked mode" during development, which enables dynamic type assertions. Once you've debugged your code, you can run it in "production mode", which is faster.

Dart code can be converted to JavaScript to run inside mainstream web browsers, but it can also run in the dedicated Dart Virtual Machine (which is supplied in the SDK). There's also a version of the *Chromium* browser called *Dartium*, which includes the Dart VM. The language includes an extensive standard library with routines to handle I/O, maths, strings, UTF8, JSON and more. So despite its youthfulness it's already a well kitted-out language with plenty going for it.

Note that Dart is used inside Flutter for application development – in other words, Flutter app developers write their code in Dart. The Flutter framework itself is written in a mixture of languages including C and C++.

Q Right. But iOS doesn't include the Dart VM, so how do Flutter apps run in it?

A Everything is converted into native code. The C/C++ components are compiled with LLVM/ Clang, while the Dart code is converted ahead of time. No interpreter is involved when the app built on Flutter is running – one of the reasons why performance is so good.

Q Can I build games or desktop apps with Flutter?

A You could try, but that's not what the framework is designed for. It's very much focused on mobile devices, providing fast 2D performance and a consistent and clean widget set via Google's much vaunted Material Design. This is a set of interface design guidelines that focus on grid-based layouts and fast animation effects, and while the widgets (buttons, sliders, text entry boxes etc) tend to look rather flat, a smattering of drop shadows and lighting effects help to make it clear which bits can be interacted with. If you have a smartphone running a recent version of Android, you've probably already seen Material Design in action – especially in Google's Mail and Maps mobile applications.

(Of course, as Flutter is open source, nothing's stopping you from hacking away at its code to make it more suitable for desktop app development. Just don't expect a great deal of help from Google.)

Q Alright, that's enough background information! Now show me some code...

A Very well. This is FAQ and not a coding tutorial, but if you want a sample of how the language looks, here's an ultra-simple program that displays a widget with the famous "Hello, world" on it:

```
import 'package:flutter/material.dart';
void main() => runApp(new Center(child:
new Text('Hello, world!')));
```

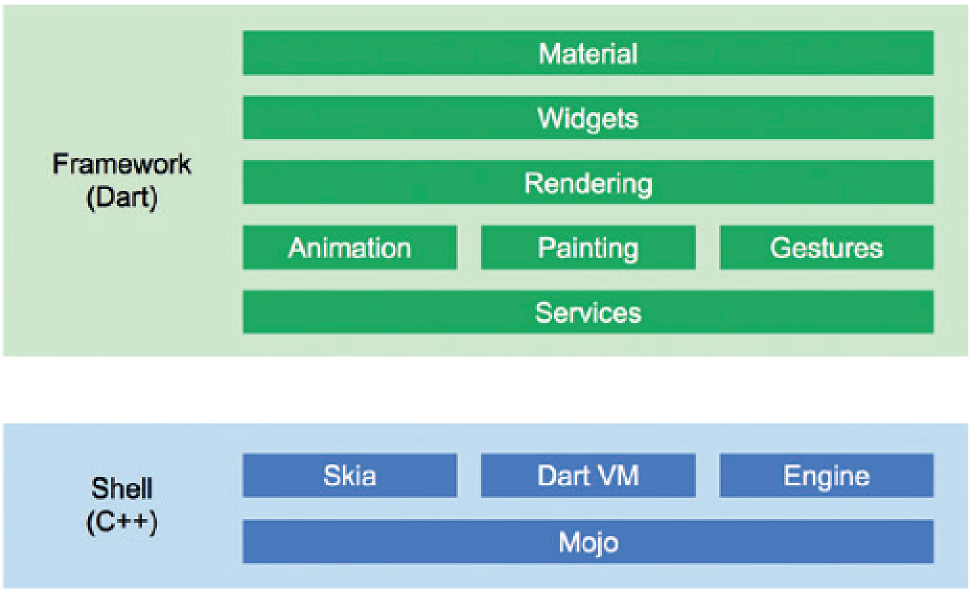
Here, **runApp** is a function that takes a root widget as a parameter. There are two widgets in use here: **Center** (which is responsible for positioning on the screen) and **Text** (which shows the "Hello, world" label.

With Flutter you can build simple applications using standalone widgets like this, or you can use more complicated Material Design widgets for snazzier-looking software. Flutter apps should look – and respond – in the same way on both Android and iOS, so you shouldn't have to worry about subtle differences messing up your screen layouts.

Q OK, sounds good. I have an idea for a mobile app that will change the world, cure the common cold and make me megabucks – where do I begin?

A Hold your horses for a moment! Flutter is still an "early stage open source project" and missing some big chunks of functionality right now. It's fun to play around with and see what it's capable of, and it's arguably worth learning early on in case it takes off as the Next Big Thing™. You'll find the main website at **http://flutter.io**, and from the instructions there you can install the SDK on Linux and Mac OS X (Windows support is planned for the near future).

The site also has a brief tutorial explaining how to arrange widgets and take user input, using a shopping cart application as an example. Otherwise the documentation is lacking in many places – hopefully this will be rectified closer to the first official release of the framework. And of course, if you ever become insanely rich one day from building apps using Flutter, please don't forget your friends at Linux Voice who introduced you to it. We wouldn't say no to a round of beers...



To learn more about the system architecture behind Flutter, check out the presentation slides at http:// tinyurl.com/flutterarch. Let us know if you'd like Linux Voice to cover Flutter and Dart in more detail.

BRADLEY KUHN

SOFTWARE FREEDOM CONSERVANCY

Graham Morrison meets someone who really does get the difference between free as in beer and free as in freedom. Mmmm, ~~beer~~ freedom!

The GPL guarantees the freedom of the Linux kernel, GCC compiler (so we can write programs to run on the kernel) and other everyday heavyweights such as the MySQL database, used by pretty much every major website. Or at least that's the theory – in reality there are lots of Bad People who don't give a flying monkey's about software freedom, who think they can just take whatever code they think will make them money and fiddlesticks to the rest of us.

Enter, Bradley Kuhn. As president of the Software Freedom Conservancy it's his job to keep an eye on corporate interests to make sure Free Software stays free. We were lucky enough to catch up with him to find out more.

LV We're sitting here at the biggest Free Software conference event of the year, O'Reilly's OSCON. But we've noticed it changing over the past 10 years or so. There was a lightning talk this year arguing against copyleft, even saying that the 'viral' nature of copyleft is damaging and that we should move forward to permissive licences. It seems that open source has reached a point where there's a generation that doesn't know what it was like before it existed, and that's a huge danger.

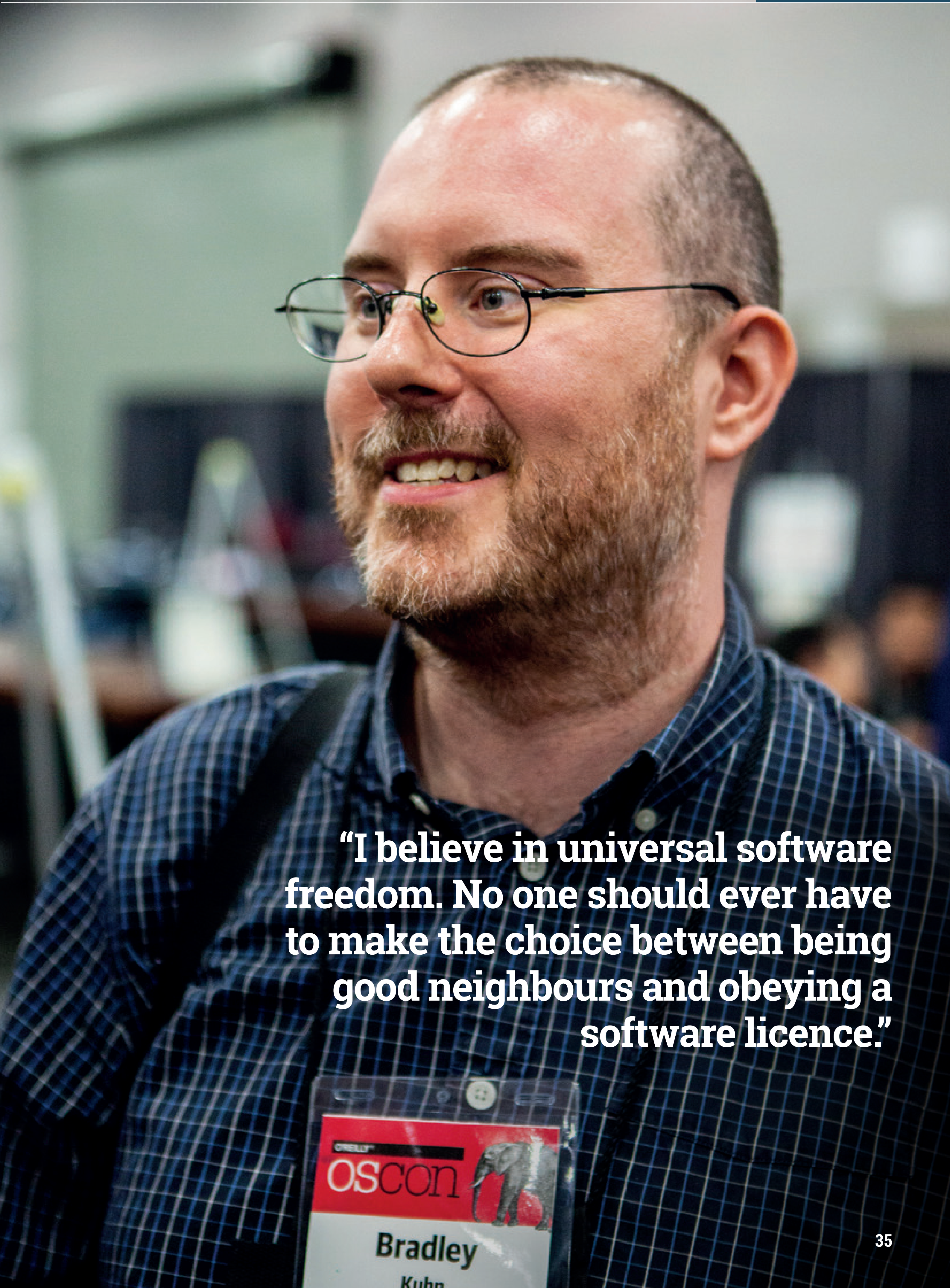
Bradley Kuhn: It's a very big danger. This is a focus and centre of my work. At this very conference in 2002 I believe, there was a situation where Microsoft first came to this conference and their messaging, after attempting to try to destroy the GPL (they had this messaging in 2001 saying that GPL is un-American, it's a virus, it's a cancer, it eats your software like a Pacman, which was Bill Gates' contribution to the debate), they decided to pursue co-option, which is what many of the for-profit companies have done with Free Software over the last decade and a half.

Microsoft was one of the first companies to try to co-opt Free Software. The way they tried to do it was to find what they saw politically as the key faultline in our community that could disrupt us, which was copyleft vs non-copyleft licensing. That's kind of the way co-option works; your opponents find the middle of the ground and say that that's the radical thing and now "we're going to embrace it". That's what we see happening here and that led to the lightning talk you saw. I've seen many attacks by the Apache Software Foundation on copyleft. They had a page up where they compared copyleft to the Business Software Alliance and so forth, and while they softened that page after I criticised it a few times in my talks, it still basically says copyleft is viral – viruses make you sick, it's the classic attack against the GPL! [Readers can see **https://www.openoffice.org/why/why_compliance.html** and **https://web.archive.org/web/20150120123334/http://www.openoffice.org/why/why_compliance.html** to see the various versions of these statements.]

At this point, I don't think I'm under any obligation to stand with a non-copyleft licence advocacy, so I've in fact changed my rhetoric on it. I'm happy to say that copyleft is better because copyleft stands up for software freedom, and basically non-copyleft licences lay down for software freedom. They give software freedom to you but they allow companies to proprietarise it.

There's no surprise that there's this push for non-copyleft by businesses. It's not a conspiracy by any means; these companies all want non-copyleft as a simultaneous spontaneous alignment of the same self interest. They all want more non-copylefted software all for their own reasons, some of which don't even overlap, but all of them are agreed that copyleft is not something they want because that requires that their own software be liberated at least some of the time. Non-copyleft leaves "all of their options

Bradley's a big fan of Richard Stallman, and worked with him at the Free Software Foundation before joining the SFC.



"I believe in universal software freedom. No one should ever have to make the choice between being good neighbours and obeying a software licence."

The SFC uses the law to protect the GPL, but it prefers to cooperate with violators to avoid the expensive business of taking them to court.

open" and it means they can consume as much as they want without giving back. And I'm not saying lots of companies participate in this anti-copyleft rhetoric.

LV Even Apple does it.
BK: Right, and even Apple are releasing some things as Free Software but they want to control completely and say "only some things are going to be Free Software and we decide – only we companies decide what gets to be Free Software". I have been working my entire career for a world where all the software is Free Software.

I believe in universal software freedom. No one should ever have to make the choice between being good neighbours (ie, sharing your software) and obeying a software licence. I was convinced by Richard Stallman's arguments 20 years ago that that was correct and I still think it's correct. It's unfortunate now that this co-option has occurred because it makes it much easier for everyone to compromise with a partially Free Software world. I compare it often to what's going on in the environmental movement. Climate change is a disaster that we're facing and it's a slow-moving time bomb.

LV Not that slow...
BK: Yes, true enough, not that slow but it is in a larger time scale; it's not going to happen in two years, it's going to happen in maybe 15 or 20. The environmental movement has had a tremendous amount of success. One of the crazy things that you see as you walk around this conference centre is all the recycling bins everywhere, and people might point to those bins and say, "Well, that's successful environmentalism".

Yet we still have climate change; there's so much work to do. And the corporate adoption of open source is exactly like recycling programs. Every company by analogy has a recycling program now, but they're ignoring the climate change version of our issues in the Free Software world.

LV People recognise it's a necessity to tackle climate change. For a long time, copyleft was a necessity; software freedom wouldn't have worked without it. And now companies don't appreciate that necessity, so they come along and change things, not in an evil way to make them do anything other than what's in their own best interests. There's nothing wrong with that, but it's a huge challenge.
BK: I agree with you. That's why I've worked in non-profit charities my entire career, because if you look at how Free Software started, it started in charities. The idea of the founding of the Free Software Foundation – where I serve on the board of directors although it's not my primary day job – was to write Free Software as a charitable activity, and the FSF did that. That's how GNU came into existence, that's why we have *GCC* and that's why we have GNU *Emacs*, and all of the stalwart packages of Free Software. Over time, companies realised that they have an interest in making money from Free Software. The earliest writings of Free Software always said 'we're going to treat all commercial and noncommercial activity equally' (not that commercial activity is more important, it's equal to noncommercial activity). Therefore people began making money with Free

> I have been working my entire career for a world where all software is Free

Software. And because their initial businesses were in copyleft communities – Cygnus [Solutions] being the classic historical example, which Red Hat eventually bought – was making money by adding changes to the copyleft software.

But once we got to the point that was this co-option moment of open source, companies just came along and they wrote the software from scratch without copyleft. And that is where the real threat comes, because companies are going to want to "keep their options open": they'll decide what they want to release and, if they do use a copyleft licence, they'll often use it, in my view, manipulatively, where they keep all the copyrights and then try to trick people into buying a proprietary licence. Copyleft was not designed to trick people into buying a proprietary licence, which is why I'm a fan of multi-copyright GPLed projects run by communities, but we're having fewer and fewer of those and that's a deep concern of mine.

LV You were in the TODO Group meeting (a group of campanies that use open source, including Google, Facebook, Twitter, HP, GitHub, Dropbox and Yahoo) earlier and made an important point about including GPL representation. These were companies whose success would never had happened without the GPL.

BK: I've seen a lot of these kinds of initiatives start up – and there's more and more coming around – for-profit companies getting together, often forming a trade association.

I think people forget that there are different types of non-profits. Trade associations form to push forward a common business interest of what companies want.

LV Such as the Linux Foundation?

BK: The Linux Foundation is indeed a trade association and it's so different to a charity. I don't have any objection to them existing. I think, actually, trade associations are a valuable thing to have in the community. But we also need charities, because charities fight for the public good. I fight for the individual user and developer, not the companies who give us money. In fact, the IRS [Internal Revenue Service] in the US, which decides who's charitable, would shut my employer – Software Freedom Conservancy – down if we did things in a company's interest. Whereas a trade association is designed to be exactly that – all the businesses get together and decide what they want, and the trade association goes out and advocates for it.

Now that these companies are anti-copyleft, there is more anti-copyleft coming from more quarters, including trade associations and for-profit companies. And meanwhile the number of people that violate the GPL continues to go up.

Sorry to go off on little a tangent, but it's important to know that copyleft only works if there's code under copyleft that someone wants to incorporate into their product so much that they're willing to give it a go and try the licence. If the code's not interesting or they can write it again from scratch, the copyleft doesn't work, because they'll just write it again from scratch under a non-copyleft or proprietary licence. In fact, that's even what's happening with *LLVM*. Apple's funding *LLVM* to rewrite a compiler from scratch – it was a university project initially with good motives and Apple's somewhat taken it under its wing so that Apple can get rid of *GCC*. And that's what

"If we don't remember the past, we're condemned to repeat it". Hang on to your software freedom…

we see happening slowly but surely. I suspect a lot of people look at that and say it's progress, and the reason they say that's progress is that there's a lot more Free Software written every day. The weird thing that happens is that – even though we wake up every day and there's more Free Software in the world than there ever has been – there's also more times in a day when it's difficult to get a particular task done without running into the proprietary software as the only solution.

A great example is the mad rush towards Slack [the communication system] – even Free Software projects use it for their internal collaboration despite the fact that Slack is proprietary software. I use IRC every day and I know its limitations and annoyances, but it's a Free Software technology. But people see a new technology and are running to this proprietary software. We've watched this over and over again. Proprietary technologies keep coming out and wooing users.

LV That's what happened with the BitKeeper version control system, which forced Linus writing his own – Git.

BK: Of course, this is repeated history and should be condemned. I look at IRC and Slack and I see *BitKeeper* and *Git*. The fact that everybody loves *Git* and it's GPLed is an irony, particularly because when you go and use GitHub most of what you're using (not *Git* itself, that's Free Software) is proprietary software – such as the bug tracker and everything on their site.

These types of problems are of great concern to me. And I keep saying "great concern" in this interview because we're facing a very difficult time for the future of Free Software and the future of copyleft. Right now, with regard to Linux's GPL, which is the strongest program out there still under GPL, it's being generally treated like the LGPL. Proprietary kernel modules are common. We have a company like VMware – against which my organisation, the Software Freedom Conservancy, is currently funding Christoph Hellwig in a litigation in Germany – is incorporating Linux into their proprietary kernel product. We're fighting that in the courts because they will not comply with the licence because, they believe they can get away with it.

LV How does a small group of people like yourselves challenge VMware?

BK: At times I wake up in the morning and think of myself as the Michael Moore of Free Software, as in his famous movie *Roger and Me* where he tries to take on big corporations hurting people. That's the glib answer.

The serious answer is that we're a non-profit charity that ask the public to support us. It took us years to raise the funds to take on VMware... [it] took a tremendous amount of fundraising effort and a tremendous amount of time talking to VMWare hoping that they would do the right thing.

LV Two years...

BK: Actually we'd been talking to VMware for about three years by the time the lawsuit was filed, it may have even been four. I think 2012 was the first time I contacted them. So there's this constant difficulty in these kinds of challenges and because we're going to have to use the courts if they refuse to comply with friendly requests and repeated offers for collaborative help.

We're going to need to raise much more money. I hate to have to give a fundraising pitch but we're a charity and in the US donations are tax deductible. If you go to **http://sfconservancy.org/supporter** you can become one of our supporters for $120 a year, and I'd ask your readers to do that.

A lot of people are employed in jobs that allow them to release at least some of their work as Free Software and it's wonderful and I'm so happy that people are doing it. But developers are also incredibly highly paid in the industrialised world. So, $120 a year is not a lot to a software developer in Europe or the US, and you can really help us fight this if you care about copyleft. You should donate to us because we're the only one defending the copyleft on Linux – the only one.

"I think [Google's] Open Source Program Office really does get it and understands Free Software."

"You can't blame Twitter too much for doing what's best for their business, because they're a for-profit company, but I can blame the general principle that we build societies around, 'Do what's best for your business rather than do good for the world,' and this is why we need charities."

LV Why do you think big companies that have made their success on the back of Free Software aren't more willing to support people like the Software Freedom Conservancy?

BK: With regards specifically to what we were just speaking about – defending copyleft – I think a few companies understand at least why copyleft needs to be enforced and what convinces them sometimes is that they spend a lot of money trying to do the right thing under GPL and their competitors don't. If you take a company like Google, for example, they spend a tremendous amount of resource on their open source program office complying with Free Software licences, as does HP and many of the other large companies. That's great for me because if they do have a violation, which they sometimes do, I know who to contact and they get it resolved quickly. People make mistakes.

> We're the only one defending the copyleft on Linux – the only one

But our goal is to get compliance, so once you come into compliance, we're happy. So these companies that do spend that money supporting the Conservancy; they understand that if we don't enforce the copyleft, they've spent all this money to comply with a licence that everybody else is just going to get away with violating.

In other words, there needs to be a regulatory body. Because of the way copyleft is designed, that regulator has to be some charity, not a governmental agency. It's not like the government's going to enforce the GPL for us.

But Conservancy is basically – because we're a public charity – the public body doing that work and we're basically the folks doing the compliance and enforcement. The problem is – and you have this in other areas of regulation – violators can smell blood in the water, right? They know the regulatory body is less funded than they are. Thus, violators work hard (in usual industries, the standard way is to lobby Congress) and to get the regulations loosened and loosened and loosened.

Many have theorised this caused the financial crisis. Throughout the 90s in the United States, the financial rules and regulations from [the Securities and Exchange Commission] were reduced and it allowed the crazy types of things like inappropriate mortgage-backed securities. Well it's very similar in any policy situation where two or three non-profit organisations – Software Freedom Conservancy, the FSF and WordPress Foundation – are the only ones enforcing the GPL.

We're three charities, we only have between us less than 50, possibly less than 40 employees, and that's the entirety of GPL enforcement in the world (at least charity-focused and public-good focused enforcement). And so when violators see that they think "we could put the squeeze on". It's easy to squeeze out 40 people and if enforcement is squeezed out, there's no GPL anymore. Because an unenforced GPL is the moral equivalent of a non-copyleft because if you don't enforce it, it's just like it was under the Apache licence to start with.

LV That's a very depressing thought.

BK: A lot of my work these days is depressing, but I'm a lifelong pessimist. My organisation as a whole, between me and my colleagues, remain hopeful as a group because I think that we can convince the public that we need to be supported.

Traditionally, the organisations that stand up for software freedom have been supported by for-profit grants. What we're seeing now is companies not wanting to give us grants because we're enforcing the GPL, which means the public needs to step up and I have faith that the public will step up and support us financially They will make a vote for GPL, a vote for software freedom via a vote with their dollars by donating to us. LV

LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM

BUY LINUXVOICE MUGS AND T-SHIRTS!

LINUXVOICE
FREE SOFTWARE | FREE SPEECH
"IT'S EASIER TO ASK FORGIVENESS THAN IT IS TO GET PERMISSION"
Rear Admiral Dr Grace Murray Hopper

shop.linuxvoice.com

REVIEWS

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

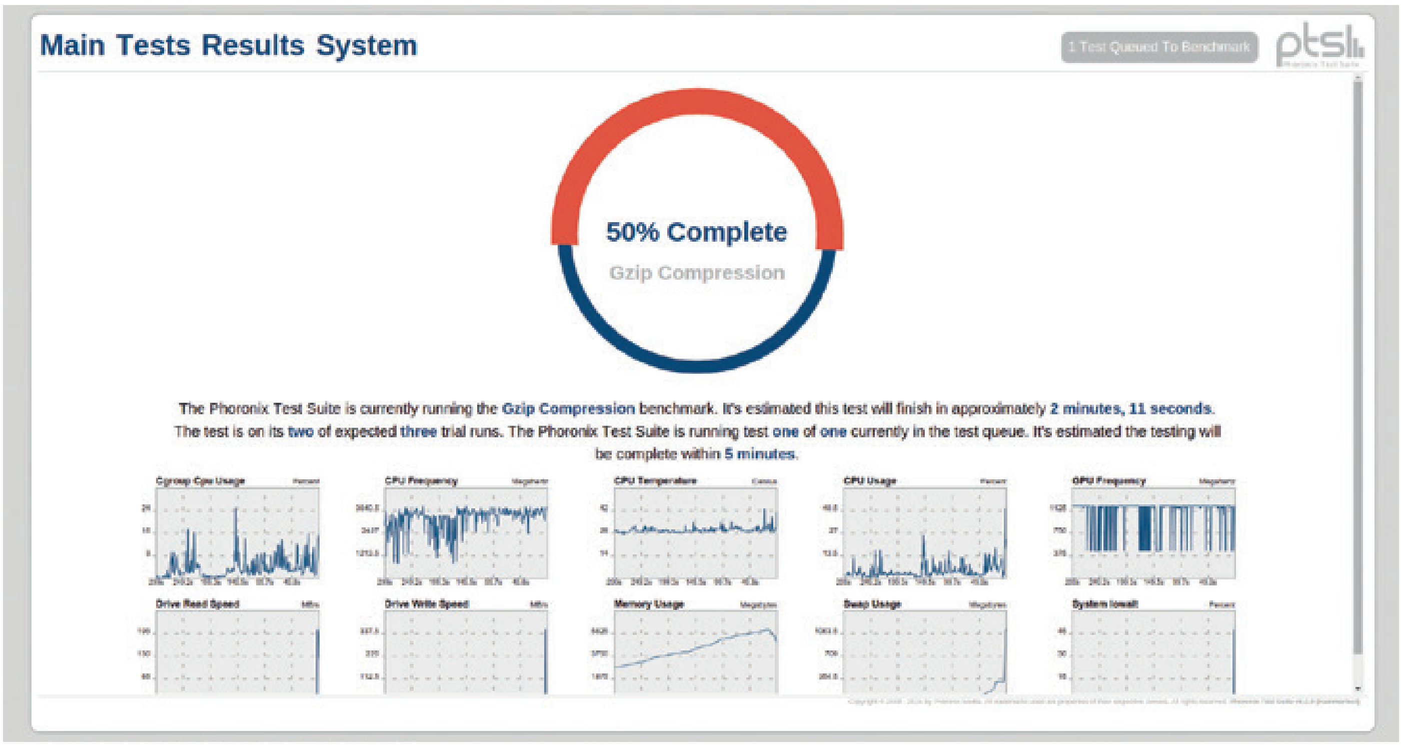
Andrew Gregory
Now has an allotment and has tasted the joys of the two-stroke petrol strimmer.

The problem with arbitrary measures of purchasing power parity is that local conditions oftem make comparisons unreliable. Take Graham's comparison of £4.00 for the Pi Zero vs £4.30 for a pint in his local pub. If I can get a drink of similar quality in my local for £2.90, does the Pi fall in attractiveness because it costs more in pints? Does it rise, because I can afford more creativity juice to inspire my projects? Or is the comparison just a bit silly?

In reality the cost to the individual consumer of the Raspberry Pi isn't the limiting factor any more. That's the time it takes to connect it up, to download, dd and install Raspbian, and to seek out worthwhile project to provide inspiration. And that's where third-party efforts, such as the CamJam Edukit, come in. £17 for a total of, maybe, 8–10 hours initial entertainment is superb value for money, and that value is entirely thanks to the imagination of the creators who put it together for you and provide the projects. The hardware doesn't matter any more – it's the people who are priceless.

andrew@linuxvoice.com

On test this issue . . .


Main Tests Results System

50% Complete

Gzip Compression

The Phoronix Test Suite is currently running the Gzip Compression benchmark. It's estimated this test will finish in approximately 2 minutes, 11 seconds. The test is on its two of expected three trial runs. The Phoronix Test Suite is running test one of one currently in the test queue. It's estimated the testing will be complete within 5 minutes.

Phoronix Test Suite 6

Ben Everard now has graphs to support his claim that he needs a new PC.

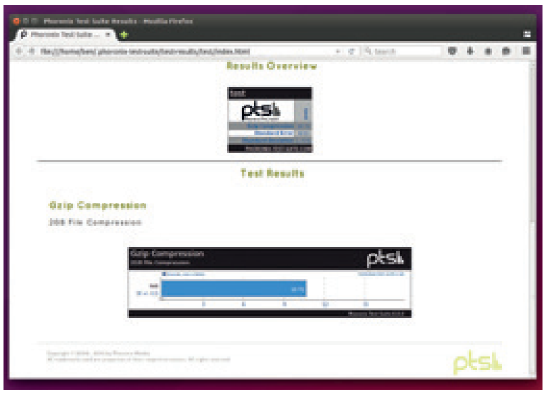
Web phoronix-test-suite.com
Developer Phoronix Media
Licence GPL v3

Phoronix is best known as the website that tests Linux hardware to find out what performs well and what doesn't. The software the team developed to perform this benchmarking – the *Phoronix Test Suite* – is open source, and has just reached version 6.

There are two interfaces to the suite: the command line interface, and the web interface. Whichever of these you use, you get the output as a HTML report that can be either saved locally or shared with the world via the **openbenchmarking.org** website. The *Phoronix Test Suite* can also combine the output of tests run on different machines to produce reports that compare the performance of different pieces of hardware. This stock of open data and the ability to combine reports means that you can easily see how your machine compares to others. If, for example, you're thinking of upgrading your CPU, you can find data on a similar machine to yours with the new CPU and compare it to your current setup before spending any money on hardware.

At present there are over 150 different tests to stress almost any aspect of your system. Tests are downloaded on the fly from the project's website, so new tests can appear at any time. These tests are grouped into categories and into suites. The latter enables you to run standard sets of tests, which is particularly useful in comparing different bits of hardware. Most of the benchmarks are based on real-world usage, such as unzipping a large file or checking the frame rate on a game.

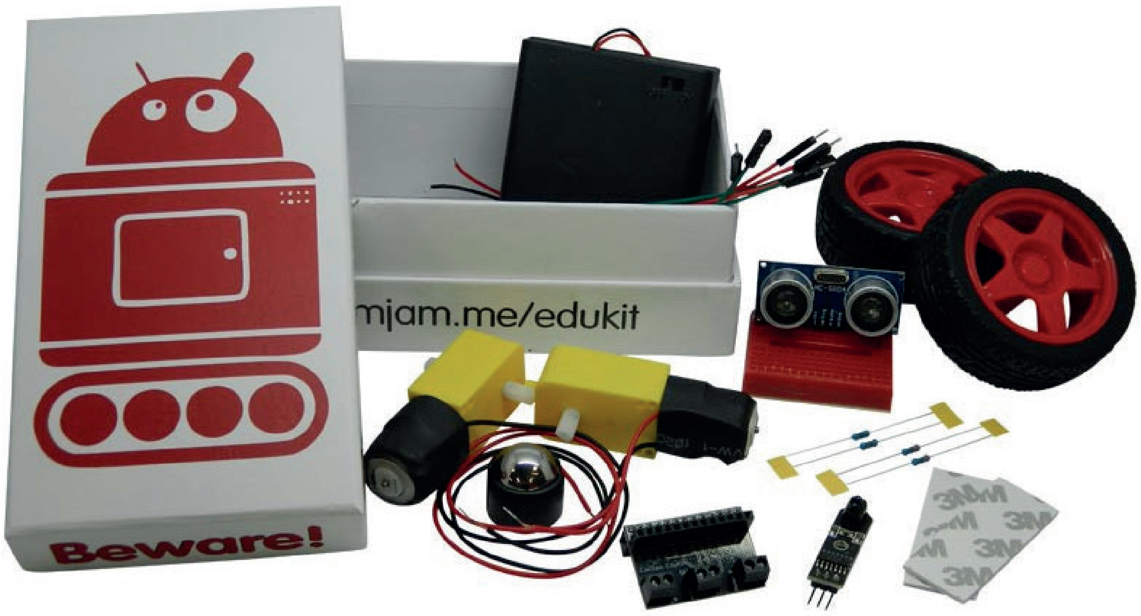
The *Phoronix Test Suite* is easy to install and use (the only major dependency is PHP, though some tests require more software to be installed), and runs on a wide variety of platforms including Linux, Windows, OS X and most BSDs. According to the project's website, even Gnu Hurd is supported, though we didn't test this. The portability enables you to benchmark different OSes as well as different pieces of hardware, and makes the suite massively useful.



The HTML reports make it easy to see the performance of your machine and compare it against others from **openbenchmarking.org**.

Easy to use, comprehensive and with a large stock of real results – the *Phoronix Test Suite* is the most useful Linux benchmarking tool.

★★★★½



CamJam EduKit 3

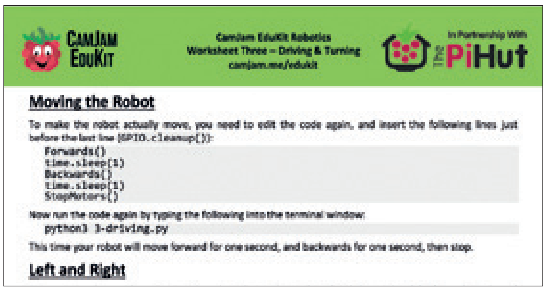
Andrew Gregory might finally get his chance for Robot Wars glory. Awooga!

You might have heard of a little device called the Raspberry Pi. It's been a massive success, but, like so many projects, the thing that's made it really take off hasn't been the device itself, but the ecosystem that's grown up around it.

One excellent grassroots exampe of this is the CamJam EduKit, so named because it's brought to you by the team behind the Cambridge Raspberry Jam. This, the third installment in the EduKit series, provides all the kit (well, most of the kit) to build a simple robot. It comprises two wheels, a ball bearing, two motors, a breadboard and a motor controller, plus the jumper cables to connect the kit to your Raspberry Pi and the required resistors. For the more advanced projects there's also a distance sensor and a black/white boundary sensor.

Ah yes, the projects. The really special bit of the EduKit is its collection of projects, available online from **http://camjam.me/?page_id=1035#worksheets**. These start from the basic Hello World that kids have to endure before they get to do anything interesting and work up to controlling motors, steering, speed and reacting to input from the sensors. We've got to take our hats off here: the directions are extremely easy to follow, and each completed worksheet leaves you having accomplished something.

The differences between Raspbian Wheezy and Jessie, coding in Nano or Idle, the several models of



CamJam EduKit Robotics
Worksheet Three – Driving & Turning
camjam.me/edukit

Moving the Robot

To make the robot actually move, you need to edit the code again, and insert the following lines just before the last line (GPIO.cleanup()):

```
Forwards()
time.sleep(1)
Backwards()
time.sleep(1)
StopMotors()
```

Now run the code again by typing the following into the terminal window:

```
python3 3-driving.py
```

This time your robot will move forward for one second, and backwards for one second, then stop.

Left and Right

Raspberry Pi and Pythin 2 vs Python 3 are dealt with smartly, rather than drowning the new user in choice.

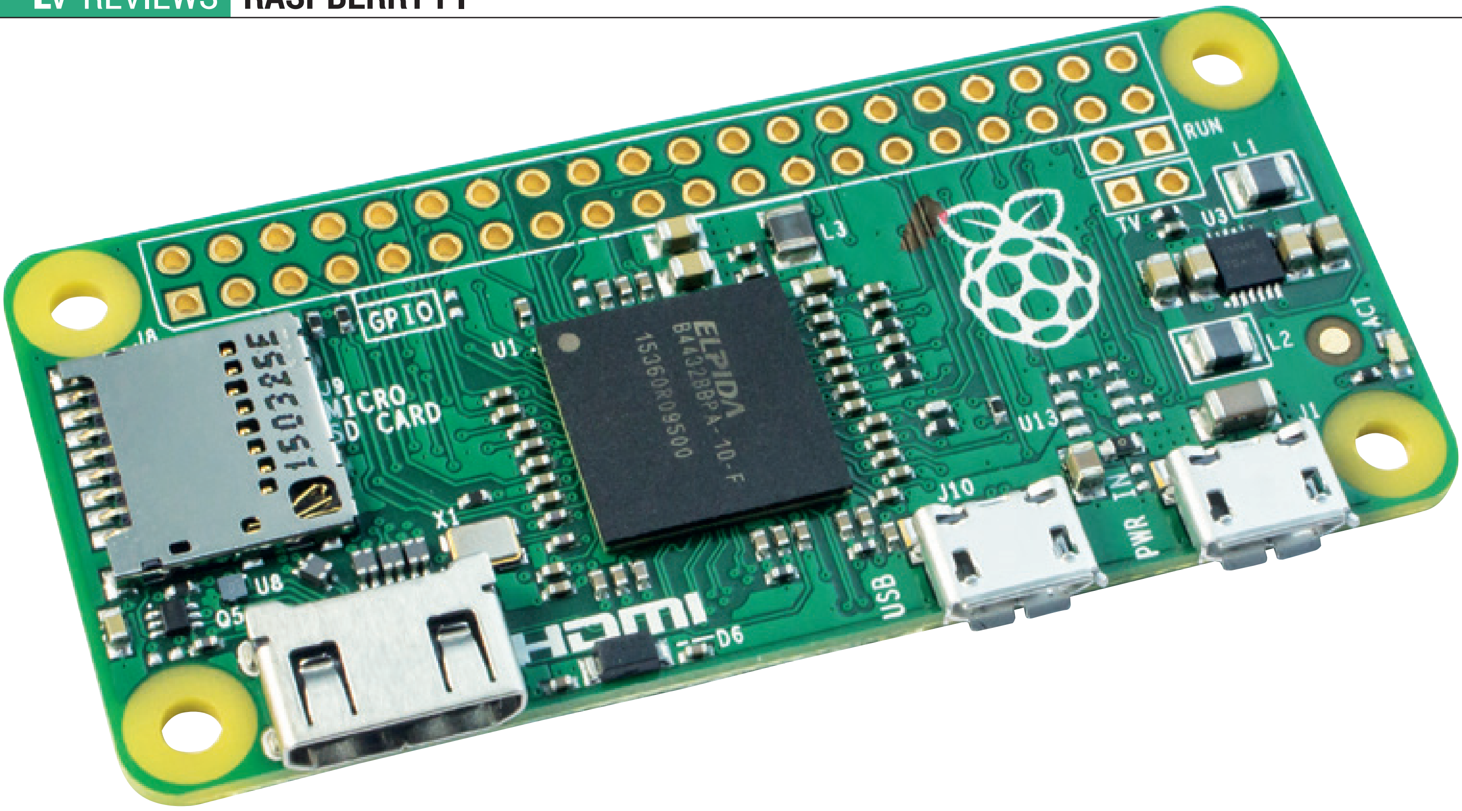
As well as programming a robot, the user is learning Linux: the **cd**, **mkdir** and **sudo** commands are introduced, so it's also a practical way to get kids into the Free Software mindset – even more so now that Daniel Bull has released the design for a 3D-printed robot chassis (the cardboard box does an equally good job), which users are free to share and modify. It's not a big deal, but it's another example of the way that building things and showing them off is so inspiring. We're jealous of the kids who are going to wake up on Christmas Day with this stocking filler. If you're looking for a way to bring a Raspberry Pi to life, this is it.

A fun, clear and inspiring resource for building your first robot army.

★★★★☆

Web camjam.me/edukit
Developer Michael Horne, Jamie Mann & Tim Richardson
Price £17 + £2 UK delivery

The worksheets still use RPI.GPIO, but plans are afoot to rewrite them using the much simpler GPIO Zero.

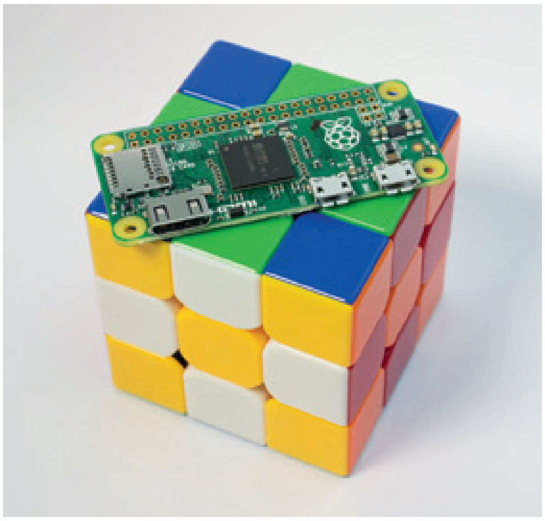


Raspberry Pi Zero

There's no such thing as a division by zero error, writes **Graham Morrison.**

Web raspberrypi.org
Developer Raspberry Pi Foundation
Price £4 including VAT

The Raspberry Pi Zero costs £4. Is there anything really left to say? For £4 you get a device that's more powerful than the original Raspberry Pi. Its CPU is factory overclocked to 1GHz and there's 512MB of RAM. It might not be the multicore ARMv7 of the Raspberry Pi 2 (not yet, anyway), but it runs everything we threw at it. Of course, compromises have been made; the camera connector has gone, as has the DSI port for connecting the official touchscreen. There's only a single micro-USB for networking and peripherals and HDMI is via a mini HDMI, rather than the fat one.


The Pi Zero is tiny (65 x 30 x 5 mm, and weighing 9 grams), yet packs more punch than the original Raspberry Pi Model B.

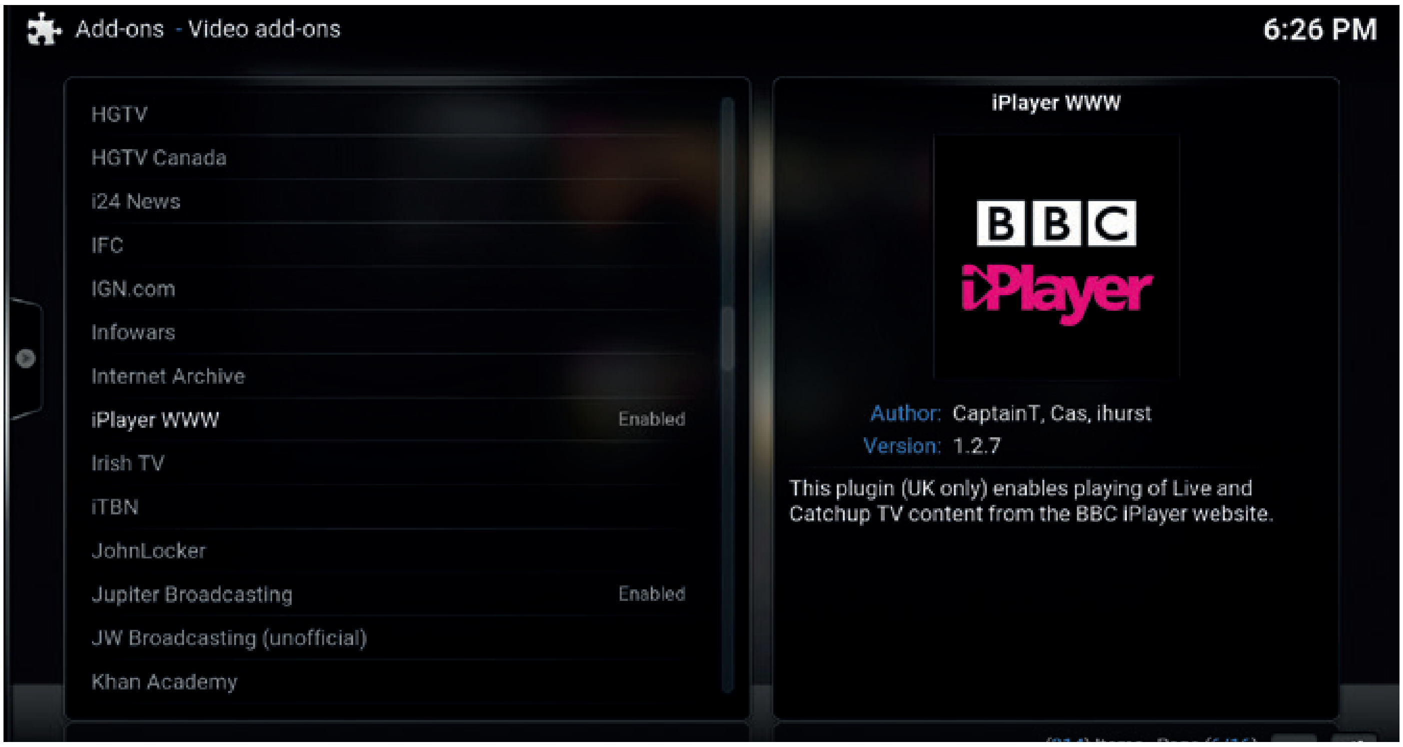
But this component diet has advantages too. The Pi Zero is tiny and light, which is something you only really appreciate when roll it over your fingers. With fewer components, power consumption is also reduced to around the 60mA mark (0.5/0.7W!). The Pi Zero could easily be sewn into clothing, secreted within a boat, or put to flight in a DIY quadcopter.

All the GPIO and composite video connections are here too, albeit unpopulated, so wires will need to be soldered manually. But at this price, the Pi becomes a genuine alternative to even the cheap Arduino copies. Our collective imagination explodes at the potential for where these devices can be placed and programmed.

If there's anything to criticise, it's perhaps the premise that a $5 Pi Zero is will make computing more accessible to people who can't afford it. Even with the original Pi, you spent more money buying the keyboard, mouse, cables , screen, Wi-Fi and storage than on the Pi itself, and if price is a focus for the Raspberry Pi Foundation, it could perhaps help bring its incredible economies of scale to these peripherals too. But really, we're only saying this to add some perspective. The Raspberry Pi Zero is a no-brainer. It's going to bring Linux and open source to tens of thousands of new makers.

We know everyone uses the price of a latte as the people's currency, but we prefer to compare the Pi Zero to a pint of Gem fine ale (£4.30).

★★★★★



Author: CaptainT, Cas, ihurst
Version: 1.2.7
This plugin (UK only) enables playing of Live and Catchup TV content from the BBC iPlayer website.

OpenElec 6

Ben Everard sets up a smart TV with a Raspberry Pi and this media centre distro.

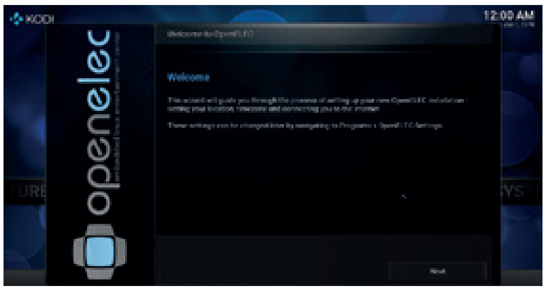
Open Embedded Linux Entertainment Centre (or OpenElec) is a Linux distro that's stripped down to just the bare essentials necessary for a home theatre PC (HTPC). It contains little more than the *Kodi* media player (version 15.2 in the latest release) and the drivers necessary to make it run.

There are a variety of images for various different hardware including x86 and a variety of ARM boards (such as the Raspberry Pi). Support for the WeTek Play TV boxes is new in version 6, as is improved support for the imx6 system on a chip that's used in the Cubiebox and Hummingboard.

When you first boot up OpenElec, you'll see a wizard that introduces the system and helps you set up the network. This means you can configure your device without a keyboard.

We tested OpenElec on a Raspberry Pi 2. This is a popular choice and a lot of work has gone into making both the underlying system and the mediaplayer run well on this hardware. The result is flawlessly smooth operation, and in our testing, we didn't come across any issues with the computing power of the hardware.

The *Kodi* media player can play local files and stream media through various addons. When we first started OpenElec, we found that there weren't any addons available to install. We had to force a refresh of the addon repository by going to System > Settings > Addons, then in the pull-out sidebar, Check



For Updates. This rather awkward process was a slight dampener on what was otherwise a really easy installation process.

There's a good range of addons including iPlayer (only in the the UK) and YouTube, but to really get the best out of OpenElec, you'll either need a digital TV tuner or a range of videos already stored on the computer that's linked to your television.

The only downside to OpenElec is that it's purely a media centre, so you can't install a regular desktop or other software alongside the media player. This shouldn't affect many people who need a media centre though, and the focus on doing just one thing well is what makes it such a highly polished, useful Linux distribution.

The best Linux media centre distro – just make sure you update the addons.

★★★★☆

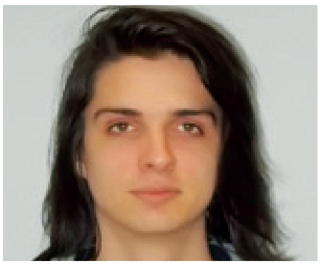
Web http://openelec.tv
Developer OpenELEC Team
Licence Various FOSS licences

Once you've transferred the image to the SD card with the **dd** command, the installation is easy even for non-technical people.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

CUTHBERT, DIBBLE, GRUB



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

With little fanfare, Valve has launched its much-awaited Steam Machines and accompanying peripherals to the masses. The fairly low-key launch was accompanied by no new major ports on launch day, which surprised many, since other announcements had been made alongside the launch of major AAA titles.

The hardware has been extremely well received. The Steam Controller in particular has received much praise, despite its steep learning curve as a result of its unorthodox design. This was expected, since its development and launch have been extremely cautious, and there is little doubt that numerous improved iterations of the hardware will appear in future.

This is really only the beginning, and clearly no one has been expecting Linux gaming to become mainstream overnight. While the positive reception will help in that regard, what remains uncertain is whether the Linux-powered consoles will see widespread success. Only a handful of Steam Machines are at price points competitive with consoles, and though better on paper, they lack the advantage of hardware-specific optimisations which give consoles their appealing price/performance ratio.

Sure, drivers are still a mess, performance is disastrous compared to Windows, but it is now out in the wild in a similar fashion to the early Android devices, and only unpredictable and irrational markets will define whether it goes the way of Android or of the Sinclair C5.

Alien: Isolation

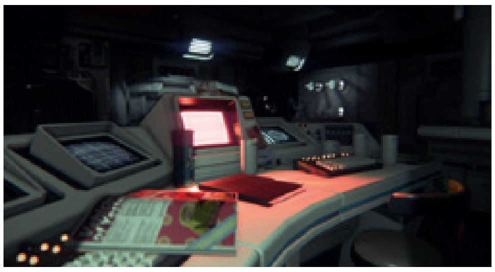
Guaranteed to raise your blood pressure.

The *Alien* franchise may have been severely overdone since the release of the original film, but *Alien: Isolation* gets back to its roots, taking place 15 years after the events of the original. The game puts the player in the shoes of Amanda Ripley, the daughter of *Alien*'s protagonist, who is now tasked with investigating her mother's disappearance.

The old-school beige panels and CRT monitor vision of the future is excellently captured, while the hollow and eerie space sounds create tension. Somehow, it also manages to pull off great AAA graphics without needing a whopper of a graphics card.

It must be pointed out that this game is often terrifying. The unpredictability of the alien, the building up of tension and the isolating environment of an abandoned space station really do leave you on the edge of your seat. The game gradually builds up the story and often lulls the player into a false sense of security and always manages to catch you off guard.

The pacing and story have been widely praised, and rightfully so. However, it is also what this game doesn't do that makes it equally as impressive. Making a terrifying game without pandering to the audiences of overacting internet personalities or descending into the realms of fan fiction are notable achievements in this day and age. Our only criticism is that while the pacing of the story is excellent, the player has to spend considerable periods hiding in lockers until the alien gets bored, and this tends to drag a little. That said, there is little to nothing else to criticise about this excellent game.



The game does well to recreate the feel of 70s and 80s sci-fi films (and Graham's synth collection).

Website http://store.steampowered.com/app/214490 **Price** £31.99

Being persistently and unpredictably stalked makes you feel constantly on edge.

The game lulls you into a false sense of security and always manages to catch you off guard

The Long Dark

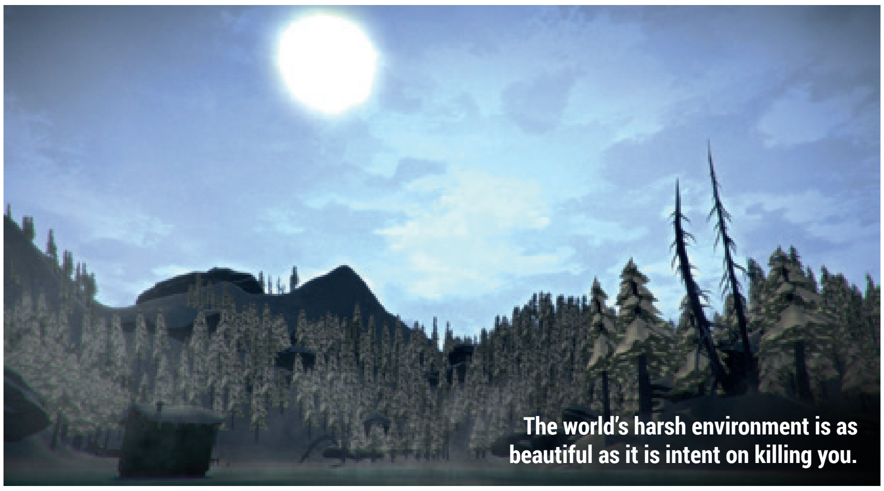
A survival game that trumps all survival games.

Despite not being finished yet, *The Long Dark* is already a deeply captivating experience worthy of purchase. When there is also the promise of an upcoming immersive story mode featuring the voice acting of David Hayter [Snake from *Metal Gear Solid*], it seems almost too good to be true.

The Long Dark does away with all the silly tree punching and crafting a GPS system from a couple of pieces of metal and some string and brings things back down to earth. This game shows how the survival genre should be approached: in the desolate Canadian wilderness, with harsh conditions and wildlife and all the unpredictability that comes with it.

There are no other players to ruin the immersion and no tutorials to hold your hand, just the full Christopher McCandless experience of the harsh struggle against nature. It's a wonder how something like this hasn't come about sooner.

Website http://store.steampowered.com/app/305620 **Price** £14.99


The world's harsh environment is as beautiful as it is intent on killing you.

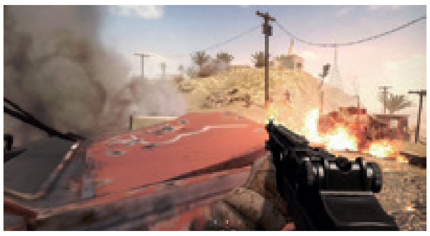
Insurgency

A team-based first person shooter that focuses on the player.

The generic maps and standard "good guys" vs "bad guys" dichotomy that plagues shooters dissuade us from trying a lot of shooters. However, multiplayer FPS games can be either excellent or atrocious based on their subtle differences, and *Insurgency* excels in this regard.

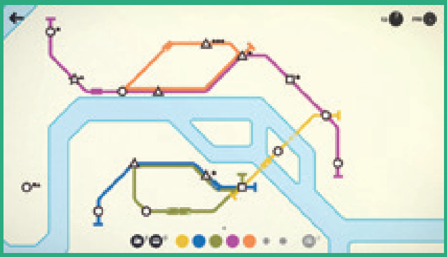
It's one of those games meant to be perfected, where every nook and cranny becomes second nature to the player and where familiarity with weapons and keyboard mappings makes the difference between life and death. However, it's accommodating in balancing realism with playability, managing to pull off realistic damage without extremely short games, since this style of play imposes caution upon the player.

The lack of kill cam is also an interesting choice, which makes maps far harder to learn while also not giving away enemy positions – adding to the realism and challenge. Getting better at the game is very rewarding, while the game modes are varied and fun without any particularly weak ones. Overall, *Insurgency* is a solid choice for those feeling trigger happy.


The source engine can often feel pretty dated, but does the job.

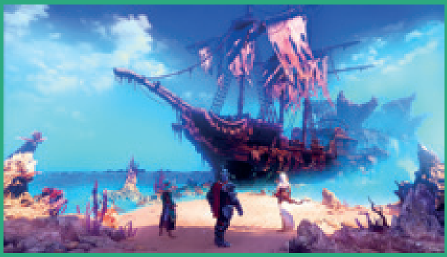
Website http://store.steampowered.com/app/222880 **Price** £10.99

ALSO RELEASED...



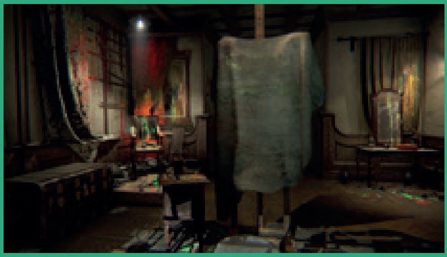
Mini Metro

Mini Metro is about as simple as they come, but it's also easily one of the most fun and satisfying experiences out there. The game presents a metro map, with an increasing number of stations appearing as the city grows. The player is tasked with building lines and making connections before the system overloads and the game ends. Ideal for a dose of fun relaxation.
http://store.steampowered.com/app/287980



Trine 3: The Artifacts of Power

The much loved puzzle-platformer series has returned, but this time with fully 3D graphics as opposed to 2.5D. The transition to a fully 3D game could have been smoother, and it's difficult to improve on itspredecessors, but it is still a solid platformer nonetheless. The game features the trademark colourful graphics that helped make the originals so appealing and still manage to make jaws drop.
http://store.steampowered.com/app/319910



Layers of Fear

This highbrow horror game puts the player in the shoes of a painter attempting to finish his greatest piece of art. Rather than focusing on jump scares, *Layers of Fear* builds up atmosphere and draws on psychological horror. The psychedelic aspects seen from experiencing the game through the eyes of madness, coupled with its knowledge of 19th century art and architecture, make the game feel very mature and pensive.
http://store.steampowered.com/app/391720

Modern Perl (fourth edition)

Can this Rossetta Stone help **Ben Everard** learn this cryptic language?

Author Chromatic
Publisher Pragmatic Bookshelf
Price £22.49
ISBN 978-1-68050-088-2

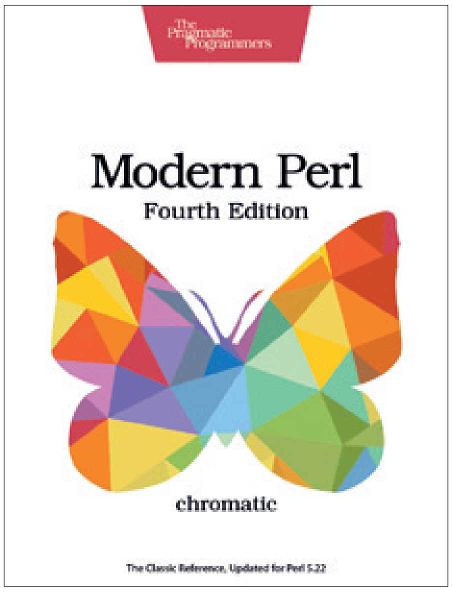
Perl can be a confusing language, even for people experienced with other languages. At first glance, it can look like a mass of symbols with little meaning, and it relies on concepts that don't feature heavily in other languages, such as context. Anyone moving to this language is likely to need a guide. *Modern Perl* is just such a guide for programmers looking to learn Perl. There's little attempt to introduce the basic concepts of programming, and the process of breaking down a problem and solving it using data and algorithms. Instead, the book focusses on Perl syntax, and the Perl language features.

As we're feeling generous, we'll call the book concise (were we in a worse mood, we'd call it sparse). The reader is taken through the concepts quickly with short examples. This works well for readers using the book alongside some existing Perl code or some challenges, but anyone hoping to read the book as a standalone guide to learning the language may be left wanting a little more.

With the impending release of Perl 6, Autumn 2015 seems an unlikely time to update a book focused on Perl 5, but if the experiences of the Python community are anything to go by, we'll see Perl 5 around for quite some time yet.

Everything you need to know about modern Perl, but too rushed to make an enjoyable read.

★★★½☆


Historians claim the modern age began in the 15th century, but Perl programmers claim modern Perl started with version 5.10 in 2007.

Practical Vim (2nd Edition)

Ben Everard isn't a real person, he's Vim script that writes Linux articles.

Author Drew Neil
Publisher Pragmatic Bookshelf
Price £19.19
ISBN 978-1-68050-127-8

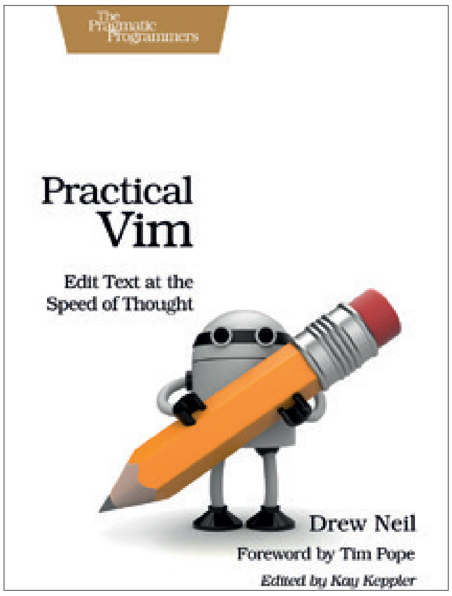
Vim is a hugely powerful editor that can dramatically speed up routine text-based tasks. All this power comes from a complex user interface that requires the programmer to get to grips with modes, commands and a myriad of keystrokes that have different effects in different modes. To get to grips with all this, you're going to need some help, and there are loads of places to go looking for *Vim* advice. The thing we liked most about *Practical Vim* when compared with other *Vim* learning materials is the smooth learning curve.

Vim is better than Emacs

Practical Vim gives the reader a nice and gentle (well, as gentle as is possible) introduction, then gradually builds up their knowledge bit-by-bit until they become a *Vim* master. Any new concepts are eased in so that the reader has time to solidify the concept in their head before moving forward. *Practical Vim* is for people who want a proper introduction to *Vim* and are prepared to put the time into learning this tool properly. If you want a quick-and-dirty 30-minute intro, look elsewhere. If you're a programmer, sysadmin, or do anything else that requires you to spend a significant proportion of your day in text files, then the time you invest in reading *Practical Vim* will be paid back handsomely in more efficient text editing.

The best way we've found to learn *Vim* other than using it for 20 years.

★★★★½


Emacs vs *Vim* is one of the oldest arguments in free software, but really we should just be grateful that there are two excellent editors to choose from.

The Martian

Graham Morrison may be the last person on Earth to to read this.

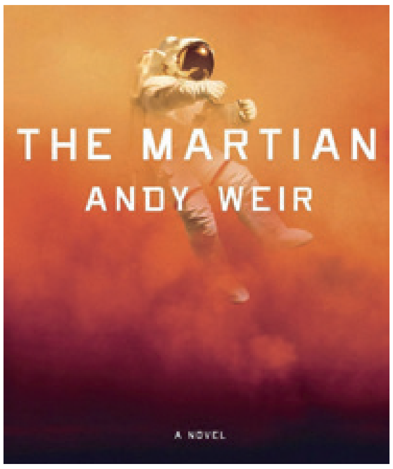
Author Andy Weir
Publisher Crown Publishing
Price £7.99
ISBN 978-1785031137

This is a science fiction story with the emphasis, mostly, on the science. We're sure you've already heard of it, thanks to the 2015 Ridley Scott film staring Jason Bourne. But it wasn't until speaking to Stuart Ward at OggCamp that we thought about the words behind the story, and we're now really grateful to Stuart for giving us this book recommendation.

The Martian is brilliant. Right from the first page, it imbibes itself with the sense of wonder at the natural world that you'd expect from a 10 year old. Everything is full of potential. No situation is insurmountable, and disasters are nothing but challenges.

The setting is obviously perfect for this. It tells the story of NASA astronaut, botanist and mechanical engineer Mark Watney, after he's stranded on the red planet Mars all alone – an embodiment of the Anthropic Principle – and it's difficult to describe anything else without giving the story away. But what makes everything so compelling is the inventiveness of both the main character and the author.

Author Andy Weir is the son of a particle physicist and obviously has a technical background. There can't be many mainstream books that contain a joke about Linux, even if we can't agree with its insinuation that Linux is more complicated than NASA's hardware. What makes the Martian so interesting is that while it's science fiction, it's most definitely set in the present or near future. All the technology described in the book is contemporary, and while it's always going to be impossible to second-guess new discoveries (the book was written before the discovery of running water on Mars, for example), *The Martian* feels remarkably authentic to us. It's obvious that the author has done an incredible amount of research into all kinds of subjects, from botany


We spent a large part of our childhood dreaming of being on a planet all alone.

to space travel, and it's this passionate engagement with science that binds the book together.

That's not to denigrate the narrative. The story is a rollercoaster ride that grows to levels of fantastic intensity, pulling you through the pages like an inescapable force. What's even more remarkable is that the book was originally self-published and given away for free. It was only after incredible demand for the Kindle version, a version that Andy Weir reluctantly sold at a minimal price in order to adhere to Amazon's conditions, that a publisher showed interest. The rest became history.

Ziggy played guitar

Even if you're not particularly into science fiction as a genre, or enjoy pretending that the robots are going to take over the world, *The Martian* is a wonderful read. It feels like how NASA should feel, had we not abandoned manned space exploration for the last 44 years. And of course, it leaves you thinking again about what might be possible were everyone to work together to create solutions to what may initially seem impossible problems, regardless of whether or not those efforts are ultimately successful.

Compelling, authentic and entertaining.

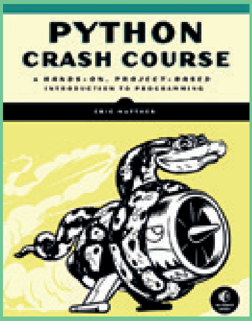
★★★★★

Also released…

February 2016

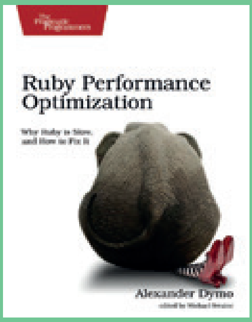
Python Crash Course

There are lots of books that teach you how to program Python. Our favourite is the 1,632-page monster that is O'Reilly's *Programming Python*. But *Crash Course* sounds good too. It's written to be quick and yield immediate results, which makes it ideal if you're looking for a primer without all the theory that often comes as part of the same read.


Python knows nothing of engine thrust hazards.

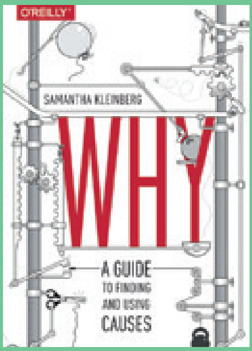
Ruby Performance

Ruby is still very important, but from the people we know using it, projects often start quickly as the developers work on their idea, before adding to the complexity, and perhaps, breaking Ruby's optimal use. Reading something about making Ruby perform better is always going to help avoid this trap, and as it's Ruby, your design is going to be better too.


Ruby's memory and CPU usage are still important.

Why

That's not one of our classic typos. This book's title doesn't have an question mark. Which could be an example of the causality it hopes to describe – whether that's studying the evidence for coffee being beneficial to your health, or what makes stock prices go up. Evidence is always good, but interpreting the evidence is much, much harder.


Why not?

GROUP TEST

Firefox and Chrome are overkill (and security risks) for many tasks. **Mike Saunders** explores the best RAM-friendly alternatives.

Lightweight browsers

Monocultures are generally very bad things in the software world, as we saw when *Internet Explorer* utterly dominated the browser market in the early 2000s. Very few innovations were being made, the browser was a huge morass of bugs and security holes, and yet Microsoft didn't seem fussed about making progress, as it had the market sewn up.

Today the situation is a lot better: we have two major open source browser projects in the form of *Firefox* and *Chromium*, plus many spin-offs from those. They're progressing quickly, a lot more reliable and secure than the *IE*s and *Netscapes* of yesteryear, and enable us to access pretty much any content on the web.

We use *Firefox* extensively at Linux Voice Towers, and it's one of the most important FOSS projects – but it's not necessarily the best tool at all times.

If you often work with primarily text-based content, where you don't need JavaScript, fancy CSS animations, plugins, webcam access and other fluff, it's well worth investigating some of the other web browsers out there. Many open source browsers are lighter on your RAM banks, they load and render pages more quickly, and because their codebases are smaller, there's a reduced risk of security holes opening up.

So we thought we'd look at six web browsers that share no heritage with *Firefox* or *Chrome* – that is, they don't use the *Gecko* or *WebKit* rendering engines. These browsers don't offer all the bells and whistles of the big names, but they do have big advantages in terms of performance and security, and have plenty of benefits for when you simply want to browse the web for useful textual content. After all, the web isn't just about cat videos, right?

These browsers don't offer all the bells and whistles of the big names, but they do have big advantages

On test

NetSurf

URL www.netsurf-browser.org
Interface Graphical
Latest release 3.3
Describes itself as "small as a mouse, fast as a cheetah and available for free."

Dillo

URL www.dillo.org
Interface Graphical
Latest release 3.0.5
Aims for the "democratisation of internet information access".

Lynx

URL http://lynx.invisible-island.net
Interface Text
Latest release 2.8.8
The classic text mode browser, dating back to 1992.

W3m

URL http://w3m.sourceforge.net
Interface Text
Latest release 0.5.3
This text mode browser sports an uncanny ability to render complex pages.

Amaya

URL www.w3.org/Amaya
Interface Graphical
Latest release 11.4.4
More than just a browser, Amaya offers integrated website editing capabilities.

ELinks

URL http://elinks.or.cz
Interface Text
Latest release 0.12pre6
A highly customisable browser extensible via Lua and Guile scripts.

Who needs pictures?

Text-mode browsers may sound pointless, but they're surprisingly useful if you want to concentrate on text-heavy content such as Wikipedia. Plus, they consume very little bandwidth, and you can use them over an SSH session on another machine.

This author was recently stuck at an airport for several hours and the awful free Wi-Fi was providing just 1 kilobyte per second. Everyone else had given up trying to browse the web, but we SSHed into a remote machine (a Raspberry Pi), ran *W3m*, and were happily reading Wikipedia and Reddit with extremely low bandwidth requirements – so it's well worth keeping a text mode browser around and learning how to use it.

A browser in your editor

Is there anything that Emacs can't do?

Ask *Emacs* fans why they love their "editor" so much, and they'll usually give you this answer: you can do (almost) everything in it, without having to leave its cosy confines. Want to check your email? Organise your schedule? Play *Tetris*? Or even talk to a virtual psychiatrist? *Emacs* lets you do all this – and much more – without having to install a bunch of different tools. We've heard that there's actually a good editor somewhere inside *Emacs*, but you have to switch to *Evil* mode first...

So it's not surprising that some enterprising hackers have created a few web browsers for *Emacs*. The most notable are *Eww* and *W3*, and they can be used in combination with *EmacsSpeak* to provide web browsing facilities for blind or visually impaired users. *W3* in particular is surprisingly featureful, supporting forms, tables and basic CSS. Another option is to integrate the *W3M* text mode browser into *Emacs*. The process for doing this, plus links to the other browsers and related information, can be found on the *Emacs* wiki at **www.emacswiki.org/emacs/CategoryWebBrowser**.

Dillo

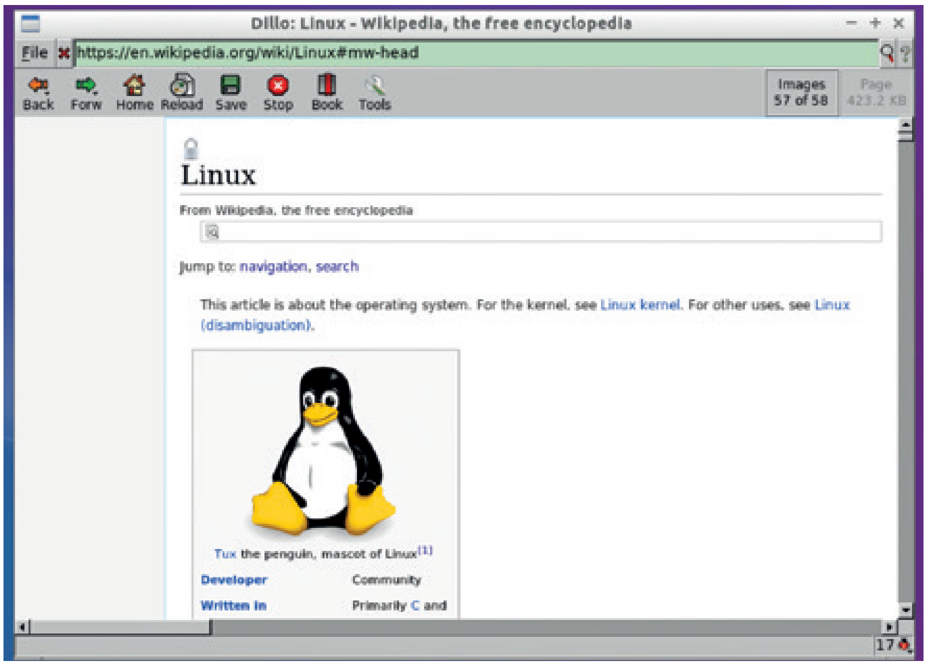
Tiny, fast and very limited.

Dillo is the first of the three graphical browsers on test, and distinguishes itself by being based on the *FLTK* graphical toolkit. This is a very light alternative to *GTK* and *Qt*, and while it looks rather unappetising out of the box, it can be themed up and does the job effectively. We compiled the latest release from source; note that you need to pass **--enable-ssl** to the **./configure** script to build in SSL support, otherwise it's disabled by default. The end result is a 745k binary which launches in a snap.

By and large, *Dillo* looks and works like a regular browser. There are familiar tool and status bars at the top and bottom of the window respectively, while Ctrl+L switches focus to the address bar and Ctrl+R reloads a page. The browser loads pages at a scorching pace, with relatively small RAM usage: when accessing the Wikipedia page for Linux, for instance, *Dillo* uses just 24MB of RAM, whereas *Firefox* eats up 235MB.

It's not obvious from the start, but tabbed browsing is possible via right-clicks on links, while facilities for bookmarking, searching for text and viewing HTML source code are included as well. And that's pretty much it – you won't find any fancy features here such as plugins or developer tools, but it has the basic browsing jobs covered adequately. Some aspects of the design are a bit annoying, however, like the lack of history in the address bar and the use of **/tmp** as the default download location.

But how does it render pages? Well, it's a very mixed bag. *Dillo*'s major problem is displaying content that's provided inside floating panes. Instead of presenting them alongside the main content, *Dillo* simply places floating elements and content after one another in one big vertical column. So in Wikipedia, for example, you have to spend ages scrolling past the Infobox panel before you reach the content. (In *Firefox*, that panel floats on the right and allows space for content on the left.)


Dillo: Linux - Wikipedia, the free encyclopedia
File
https://en.wikipedia.org/wiki/Linux#mw-head
Back
Forw
Home
Reload
Save
Stop
Book
Tools
Images 57 of 58
Page 423.2 KB
Tux the penguin, mascot of Linux[1]
Developer
Community
Written in
Primarily C and
17

Linux

From Wikipedia, the free encyclopedia

Jump to: navigation, search

This article is about the operating system. For the kernel, see Linux kernel. For other uses, see Linux (disambiguation).

Dillo is fairly good for browsing Wikipedia, although you have to scroll past elements that would otherwise be floating at the sides of a page.

Your mileage may vary

This causes issues with many websites – although it doesn't render them unusable. You just have to get used to scrolling past chunks of pages that you'd normally ignore. *Dillo* has a smattering of CSS support, but no JavaScript, so any pages dependent on that will break.

Ultimately, *Dillo* is acceptable for browsing text-heavy websites with minimal snazzy effects, like Wikipedia and Reddit. It's delightfully fast when it works, and very conservative with RAM, making it a good choice when you're reviving an old netbook and just want to use the browser for reading purposes. Unfortunately, though, it's simply too broken on many websites to make it suitable for daily general-purpose usage.

But! Support for floating elements will be included in *Dillo 3.1*, along with many other CSS attributes, which will make the browser far more usable for the majority of websites out there.

VERDICT

The best graphical choice for ultra-low RAM usage, but has major issues rendering pages.

★★⯪☆☆

Lynx

The original (but not best) text mode browser.

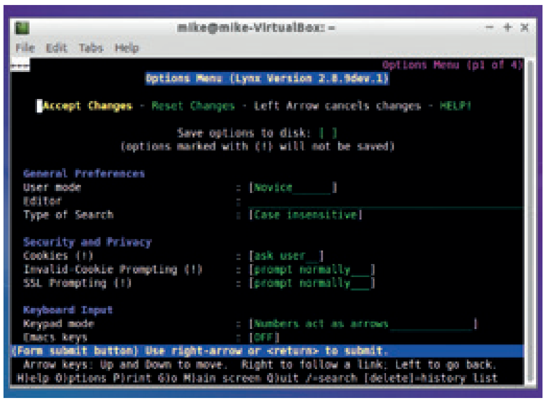
And so we come to our first browser that runs entirely in the terminal. *Lynx* has been in development since 1992, so it's pretty much as old as the World Wide Web itself, and has been ported to a vast range of platforms including MS-DOS, VMS, AmigaOS, BeOS, OS/2 and others. Although *Lynx* is still under development, progress is sometimes slow and often the project goes half a year or more between developer snapshots.

Out of the box, *Lynx* is welcoming and easy to use: a status bar and some lines of help text along the bottom provide information on common keyboard shortcuts, so you don't have to delve into manual pages just to get basic navigation sorted out. The left and right arrow keys work like backwards and forwards respectively – and to navigate to the next link on the page, use Tab. Colour is used extensively, although the default scheme often lends to garish combinations that are hard to read.

Lynx supports HTTPS and makes a decent attempt at recreating page layouts, but it's the weakest of the text mode browsers in our Group Test in this respect. It's light on RAM, using just 15MB to render the Wikipedia page for Linux, and can be operated automatically using keystrokes recorded in a file – useful for performing tests on websites.

All things considered, *Lynx* is the most mature text mode browser out there and is pleasingly user friendly given its age. With support for HTTPS and forms it can be used with an impressive range of sites, but its rendering abilities for tables and complex layouts fall behind those of *W3m* and *ELinks*.

Lynx is light on RAM, using 15MB to render the Wikipedia page for Linux


```
mike@mike-VirtualBox: ~
File  Edit  Tabs  Help
                                                    Options Menu (p1 of 4)
                    Options Menu (Lynx Version 2.8.9dev.1)

    Accept Changes - Reset Changes - Left Arrow cancels changes - HELP!

                        Save options to disk: [ ]
                  (options marked with (!) will not be saved)

  General Preferences
  User mode                               : [Novice      ]
  Editor                                  :
  Type of Search                          : [Case insensitive]

  Security and Privacy
  Cookies (!)                             : [ask user  ]
  Invalid-Cookie Prompting (!)            : [prompt normally___]
  SSL Prompting (!)                       : [prompt normally___]

  Keyboard Input
  Keypad mode                             : [Numbers act as arrows              ]
  Emacs keys                              : [OFF]
(Form submit button) Use right-arrow or <return> to submit.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```
Handily, most *Lynx* options can be configured inside the browser, so you don't need to faff around with config files.

VERDICT

Mature, accessible and easy to tweak – but lagging behind.

★★★☆☆

Amaya

A website editor and browser combined into one app.

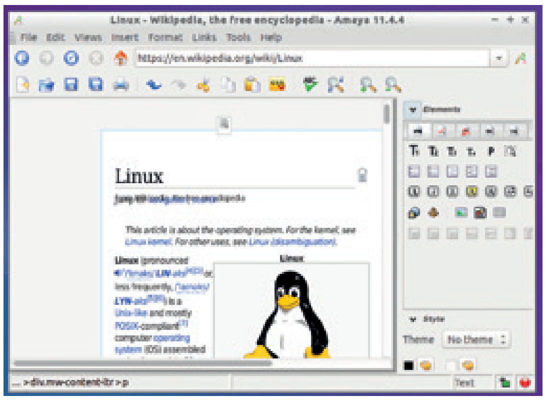
Amaya is another of those projects with an extensive history, going back to 1996. It was originally created at INRIA, the French Institute for Reasearch in Computer Science and Automation, and is still seeing the occasional update today – although the latest stable release was issued back in 2013. Just getting *Amaya* installed can be a headache; many distros, including Debian and Ubuntu don't have it in their repositories, so you have to grab the package from *Amaya*'s website and hope for the best. The app depends on the *wxWidgets* toolkit, which in turn depends on *GTK 2*, so there's a large layer of older dependencies involved.

When loaded, *Amaya* presents an information page, which you can start editing straight away. On the right-hand side you'll see a panel containing various buttons to add HTML elements – under the Views menu you can disable this in order to use *Amaya* purely as a web browser. And its performance here is a real mixed bag. *Amaya* uses OpenGL to render pages, and we encountered a startling number of bugs and glitches when trying to browse simple pages.

Sub-prime performance

This could be down to mismatches between OpenGL libraries of today and those when the packages were built, but then *Amaya* needs to generate updated packages (or work with distros to get the software into repositories). Rendering of layouts wasn't too bad when the browser was behaving, although still uses 173MB of RAM to display the Wikipedia Linux page.

So we can't really recommend *Amaya* for either of its tasks – website editing or browsing – in its current form. *BlueGriffon* and *SeaMonkey Composer* do better jobs with the former, and the latter simply isn't usable on a day-to-day basis until the freezes and rendering glitches are sorted out.


Linux - Wikipedia, the free encyclopedia - Amaya 11.4.4
File Edit Views Insert Format Links Tools Help
https://en.wikipedia.org/wiki/Linux
Linux
Elements
Style
Theme No theme
Text

One error *Amaya* gave us was "No pincher, please call crStateSetCurrentPointers() in your SPU". Er, OK...

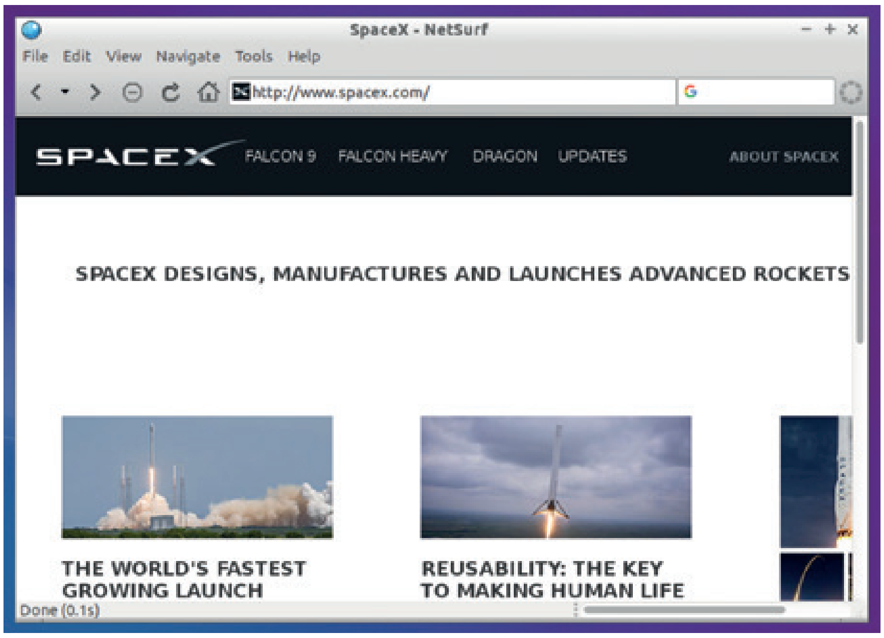
VERDICT

Has potential, but eats up too much RAM for its browsing capabilities.

★★☆☆☆

NetSurf

The most promising graphical browser.



SPACEX DESIGNS, MANUFACTURES AND LAUNCHES ADVANCED ROCKETS

THE WORLD'S FASTEST GROWING LAUNCH

REUSABILITY: THE KEY TO MAKING HUMAN LIFE

NetSurf's layout engine isn't perfect, so expect the odd glitch here and there, but it does an impressive job given the small team working on it.

Remember RISC OS, the operating system used on the Acorn Archimedes range of computers, and later the RiscPC? But back in the early 2000s, some of its users were despairing about the state of web browsers on the platform. So they created a new browser from scratch – *NetSurf* – which has since been ported to Linux.

We opted to build *NetSurf 3.3* from scratch, but the documentation inside the source tarball described a rather complicated procedure that ended up retrieving the latest code from the project's *Git* repository and compiling that instead, so we ended up with a *NetSurf 3.4* development snapshot.

Once built, *NetSurf*'s 5MB binary started up in less than a second, presenting an attractive *GTK* interface. We were happy to find that most commonly used keybindings had been implemented – eg Ctrl+L to switch to the address bar – but some behaved oddly or didn't work at all.

Hey, good lookin'

NetSurf's features include bookmarks, address bar history, page source viewing, popup and ad blocking, Do Not Track header sending, and proxy server support – so it's a well-featured lightweight browser that provides the bulk of functionality required for most tasks. We found its performance to be largely very good; not as fast as the text-mode browsers, but still better than the heavyweights on many sites. In terms of RAM consumption it's also worlds ahead of *Firefox*: where that browser requires 235MB to view the Linux Wikipedia page, *NetSurf* does it in just 74MB.

Of course, none of this matters if the layout engine is pants, but *NetSurf* is by far the best on test here. It has problems with some complex sites and struggles with certain layouts, but for the most part it does a decent job handling HTML and CSS – far better than *Dillo*. We had little luck getting JavaScript-heavy websites to work, but those sites that scaled gracefully for non-JS-enabled browsers were mostly usable.

We've become big fans of *NetSurf*: it's by a long shot the most promising non-*Gecko*/*WebKit*/*Blink* browser out there, and while it still falls apart on very complex pages, for most text-centric sites it's fast, reliable and a joy to use. Oh, and it's a great choice on the Raspberry Pi.

VERDICT
As close to a *Firefox* as you'll get, but using a fraction of the RAM.
★★★★½

Servo: the future?

Mozilla's new rendering engine.

Website rendering engines such as *Firefox*'s *Gecko* or *Chrome*/*Chromium*'s *Blink* (formerly *WebKit*) are insanely complicated pieces of software. They need to handle HTML, CSS, JavaScript, images, sound files, embedded videos and much, much more. Quite often, by the time that such a rendering engine reaches maturity, technology has moved on so much that the engine itself is rather dated. *Gecko* and *WebKit*, for instance, were started in an age when most people were using single-core computers. Today, even low-end smartphones boast quad-core chips – something that still seems crazy to us, but technology moves fast.

So *Servo* is an attempt by the Mozilla project, the makers of *Firefox*, to create a new rendering engine that makes better use of multi-core chips. It's designed for better parallelism – so that HTML parsing, rendering, layout, JavaScript processing, images etc. can all be handled by different tasks at the same time. *Servo* is written in Rust, the new-ish programming language also from the good folks at Mozilla, and while it's still in the early stages of development, Samsung has shown interest in running it on its mobile devices.

It may be a few years before we see *Servo* adopted in mainstream browsers like *Firefox*, if ever, but the end result could be faster, more stable and more secure – especially on smartphones and tablets. For more information, including access to the source code, visit the project's website at **www.servo.org**. As soon as *Servo* is ready for day-to-day use, we'll give it a thorough going-over in Linux Voice.

Servo, the parallel browser engine - Servo
Servo, the Parallel Browser Project
Getting Started
Prerequisites
Building

Servo currently runs on Linux, Mac OS X, Android and Firefox OS.

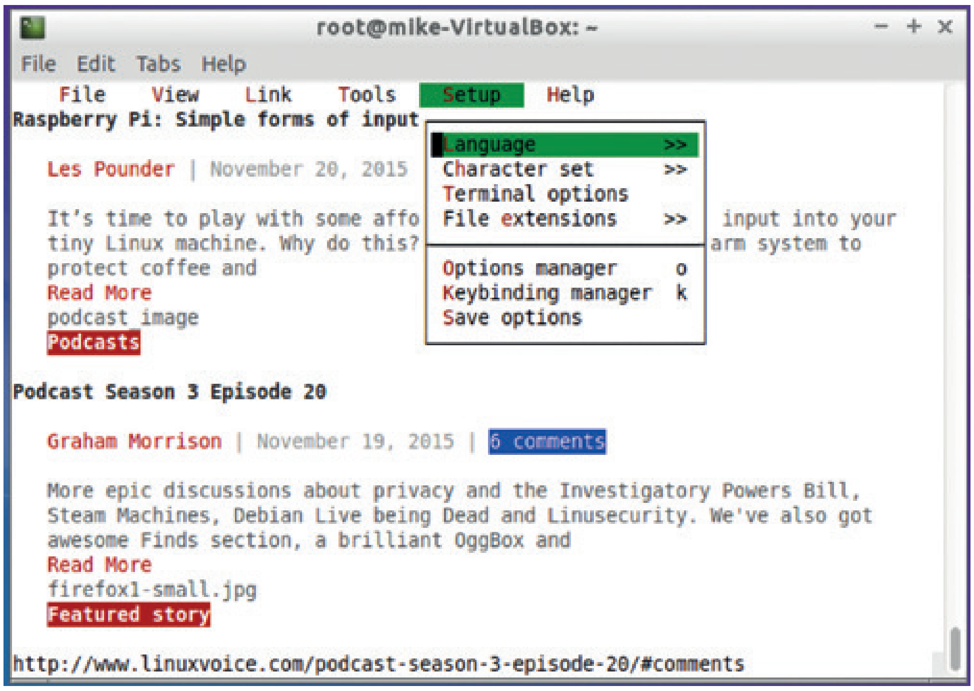
ELinks vs W3m

Kings of text mode go head-to-head.

ELinks is a derivative of the *Links* browser (which dates back to 1999), and packs an impressive amount of functionality given the limitations of text mode. The project is currently seeing very little development, though; the last stable release arrived in 2009, and most distros are using the 0.12pre6 development snapshot which is still rather old (October 2012).

The browser's most notable feature is its menu-driven interface. Hit G to open a URL dialog box, and F10 or Esc to show the menu bar at the top, then navigate through it using the cursor keys. This use of menus and dialogs makes *ELinks* feel to some extent like a graphical browser – albeit without any pictures. The Options Manager (under the Setup menu) illustrates just how capable the browser is, providing a wealth of options for for handling bookmarks, cookies, MIME types and the user interface.

ELinks supports tabbed browsing a smattering of CSS and JavaScript (if *SpiderMonkey* support is compiled in), and the ability to edit text boxes in an external editor – a lovely feature if you want to do all your text poking work in *Vim*. Performance-wise it's OK, using 16MB of RAM to render the Wikipedia page for Linux, although we found it sluggish when testing on a Raspberry Pi compared to *Lynx* and *W3m*. *Elinks*, however, handles complex layouts far better than Lynx, and does a decent job trying to get colours right as well.



```
root@mike-VirtualBox: ~
File  Edit  Tabs  Help
   File     View     Link     Tools    Setup    Help
Raspberry Pi: Simple forms of input
                                        Language            >>
   Les Pounder | November 20, 2015      Character set       >>
                                        Terminal options
   It's time to play with some affo     File extensions     >>   input into your
   tiny Linux machine. Why do this?                             arm system to
   protect coffee and                   Options manager     o
   Read More                            Keybinding manager  k
   podcast_image                        Save options
   Podcasts

Podcast Season 3 Episode 20

   Graham Morrison | November 19, 2015 | 6 comments

   More epic discussions about privacy and the Investigatory Powers Bill,
   Steam Machines, Debian Live being Dead and Linusecurity. We've also got
   awesome Finds section, a brilliant OggBox and
   Read More
   firefox1-small.jpg
   Featured story

http://www.linuxvoice.com/podcast-season-3-episode-20/#comments
```

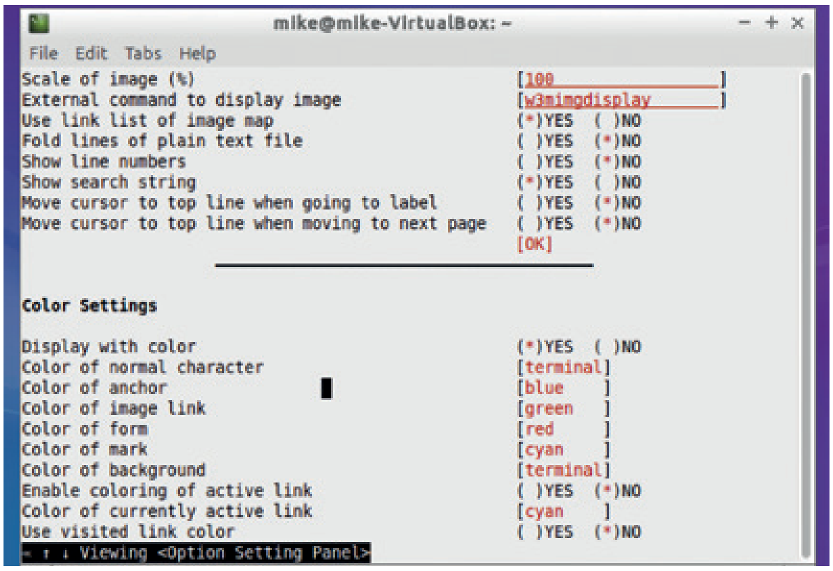
By default, *ELinks* doesn't display any colours; go to Setup > Terminal Options to fix this.

The W3m alternative

W3m provides some healthy competition to *ELinks*, but it also suffers from a sluggish pace of development – its last stable release was in January 2011. Since then there have been a couple of minor forks and patch sets, but most distros still include that old release. Still, it's not all doom and gloom: *W3m* is up there with ELinks in terms of its HTML layout engine, rendering some websites even better (but falling slightly short on others).

W3m is less accessible than *ELinks* in that it doesn't have a familiar menu-driven interface. Instead, you tap O to open a new website (ie enter a URL), and B to go back in your history. Like *Elinks*, *W3m* supports tables, frames, SSL connections and other essential browsing features – but no JavaScript.

Memory-wise, *W3m* is a stellar performer, consuming just 11MB with the Wikipedia Linux page loaded. On the whole, both browsers are very close in terms of overall features and layout engine capabilities, but *ELinks* just takes the lead with its user-friendly interface and support for a limited subsection of JavaScript, though *W3m* has the advantage of lower memory usage and slightly faster page rendering.



```
mike@mike-VirtualBox: ~
File  Edit  Tabs  Help
Scale of image (%)                                       [100            ]
External command to display image                        [w3mimgdisplay  ]
Use link list of image map                               (*)YES  ( )NO
Fold lines of plain text file                            ( )YES  (*)NO
Show line numbers                                        ( )YES  (*)NO
Show search string                                       (*)YES  ( )NO
Move cursor to top line when going to label              ( )YES  (*)NO
Move cursor to top line when moving to next page         ( )YES  (*)NO
                                                         [OK]
                    ____________________________________

Color Settings

Display with color                                       (*)YES  ( )NO
Color of normal character                                [terminal]
Color of anchor                                          [blue    ]
Color of image link                                      [green   ]
Color of form                                            [red     ]
Color of mark                                            [cyan    ]
Color of background                                      [terminal]
Enable coloring of active link                           ( )YES  (*)NO
Color of currently active link                           [cyan    ]
Use visited link color                                   ( )YES  (*)NO
← ↑ ↓ Viewing <Option Setting Panel>
```

W3m has a GUI-like options screen, but we prefer *ELinks*'s menu and dialog-driven approach.

VERDICT

ELINKS User-friendly and feature-packed – the best text mode browser you can get.
★★★★☆

W3M Up there with *ELinks* in terms of its layout engine, but harder to learn to use.
★★★☆☆

OUR VERDICT

Lightweight browsers

So, after examining six lightweight web browsers that all eschew the popular *Gecko* and *WebKit*/*Blink*/*KHTML* rendering engines in favour of their own home-brewed ones, what have we learnt? Well, one thing is clear: none of them are full-time replacements for *Firefox* or *Chrome*. But that's not a bad thing. Different tools should be used for different jobs – *Vim* is awesome for programming, but for writing a shopping list you're more likely to use *Nano*. *Gimp* has oodles of features for professional image processing, but if your kids want to draw a picture, then *TuxPaint* is the far better choice.

That's how we should look at these browsers. *Firefox* and *Chrome* may be the most sensible choices for browsing complex websites packed with CSS effects, JavaScript, HTML 5 media, plugins and so forth, but they're heavy beasts with a big attack surface for security vulnerabilities. For light browsing of text-oriented sites such as Wikipedia and Reddit, using one of the browsers in this Group Test makes a lot of sense. They load pages quickly, use hardly any RAM in comparison, and are way more suited to low-spec devices such as the Raspberry Pi.

So what's the winner? For the graphical browsers, it's *NetSurf* by a long shot. Its rendering engine occasionally leaves a lot to be desired, but it's still way ahead of *Dillo* and *Amaya* and usable on most websites that we tested. Plus, it's undergoing active development and we can hopefully expect a 3.4 or 4.0 release in the near future.

Over in text-mode land, *ELinks* takes the crown here thanks to its menu-driven interface, rudimentary JavaScript support, and impressive rendering capabilities given the restrictions of text mode. It's sometimes a bit sluggish on the Raspberry Pi, pausing for several seconds when loading large pages – an area where *W3m* does better (and also in memory usage). Indeed, *W3m* renders some pages even better than *ELinks*, so it's worth keeping both around, although it takes more time to learn its keybindings and general operation.

> NetSurf is way ahead of Dillo and Amaya and usable on most websites that we tested

Browse without a web browser

Believe it or not, you can browse the web without even using a web browser. Well, when we say "browse" we mean you can view the HTML source of a website. And you can't follow links unless you type them in manually. And don't even think about CSS or JavaScript...

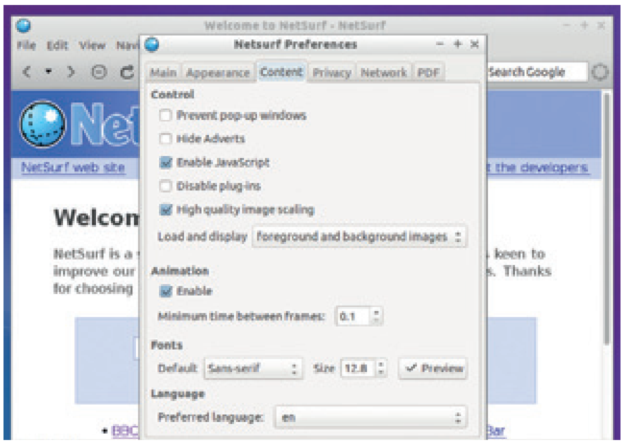
But still, it's a fun trick. Every Linux and Unix flavour includes a command line tool called Telnet, which establishes a connection on a remote machine on a specific port, and then sends textual data to it. When a web browser connects to a website, it usually connects to port 80 and says "Hey, send me some HTML please". Try this:

```
telnet lynx.invisible-island.net 80
```

A few lines of text will appear. Type in this, pressing Enter twice afterwards.

```
GET / HTTP/1.0
```

A bunch of HTTP header lines will appear followed by the HTML source for the *Lynx* website. There you go – you are "browsing" using nothing more than a tiny tool included with every Unix-ish OS out there.



Control
- [ ] Prevent pop-up windows
- [ ] Hide Adverts
- [✓] Enable JavaScript
- [ ] Disable plug-ins
- [✓] High quality image scaling

Load and display: foreground and background images

Animation
- [✓] Enable

Minimum time between frames: 0.1

Fonts
Default: Sans-serif Size: 12.8 ✓ Preview

Language
Preferred language: en

NetSurf provides a browsing experience close to *Firefox* and *Chrome*, albeit with a few issues on some complex pages.

1st NetSurf

Killer feature Rendering engine
www.netsurf-browser.org
Provides the bulk of functionality you expect in a browser, and handles simple sites with aplomb (and using little RAM).

2nd ELinks

Killer feature GUI-ish design
http://elinks.or.cz
Impressively feature-rich for a text-mode browser, and a great help when you're SSHed into a box and need to browse the web.

3rd W3m

Killer feature Speed daemon
http://w3m.sourceforge.net
Ultra fast and lightweight, *W3m* is a superb alternative to *ELinks* and renders some sites even better.

4th Dillo

Killer feature Active development
www.dillo.org
Lags behind *NetSurf* in terms of its layout engine, but the next release should be a huge improvement here.

5th Lynx

Killer feature Ultra maturity
http://lynx.invisible-island.net
It's as old as the hills and supremely reliable, but can't compete with *ELinks* and *W3m* on complex websites.

6th Amaya

Killer feature Page editing
www.w3.org/Amaya
Very glitch-prone on s Linux distros, but could be a useful website editor if the bugs were fixed.

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

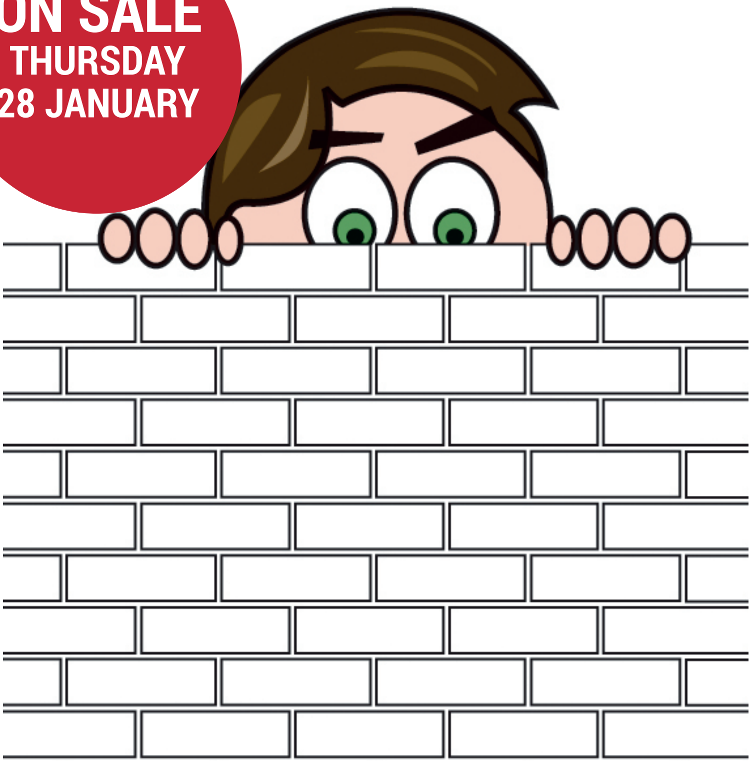
Access our rapidly growing back-issues archive – all DRM-free and ready to download

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

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

NEXT MONTH IN LINUXVOICE

ON SALE THURSDAY 28 JANUARY



SNOOPER'S CHARTER

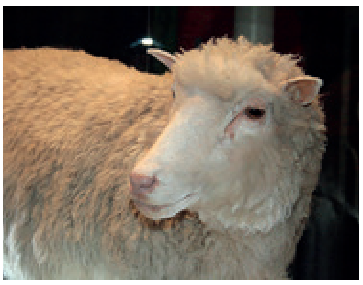
Or: why the government is wrong to even attempt to implement mass surveillance on its subjects. Free Software to the rescue!

EVEN MORE AWESOME!

MariaDB

MariaDB

It's the database that's powering an increasing number of sites in an increasing number of distros. Learn its ways and harness its awesome power!



Cloning

No, not Dolly – we're talking about cloning machines over a network, whether that's in a big business or just helping your dad out with his laptop.

OnlyOffice

Easy cloud email, calendaring and document sharing, untainted by the grasping fingers of Google's advertising wallahs. Sounds good to us.

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 August 2016 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2015
ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**

FOSSpicks

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

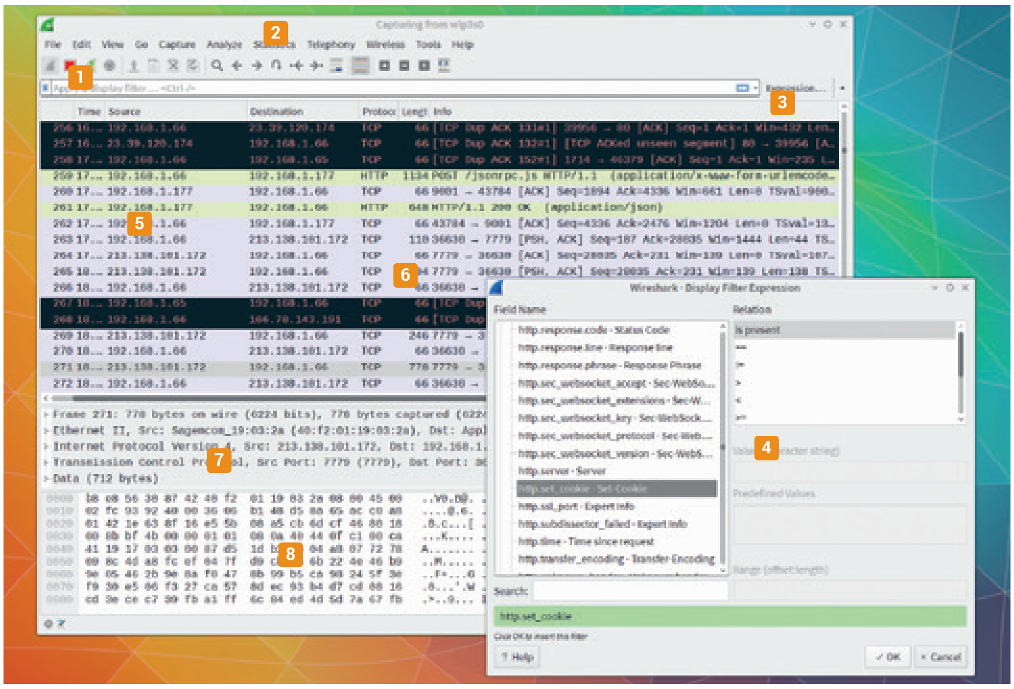
Out benevolent editorial overlord **Graham Morrison** tears himself away from updating Arch Linux to search for the best new free software.

Network monitor

Wireshark 2.0

Wireshark has been around for long enough to become both an exceptional security privacy tool and a prerequisite installation for any system administrator. It gives you the ability to take a deep dive into the packets travelling across your network (or more accurately, through the device that *Wireshark* is connected to), letting you analyse them for things like their type and content, as well as their timings, sources and destination. It understands the innards of hundreds of different protocols, and it can even decrypt protocols such as Kerberos, SSL/TLS, WEP and WPA2. As long as digital data passes between two devices, *Wireshark* will be able to capture and interpret the data, presenting it as a contextualised list waiting for your filters and analysis.

Wireshark 2.0 is a major update that has been brewing for some time. In particular, there's a new user interface built around *Qt* to strengthen its cross-platform credentials.



Capturing from wlp3s0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
1
2
3
4
5
6
7
8

```
     Time  Source            Destination       Protocol Length Info
256 16... 192.168.1.66      23.39.120.174     TCP      66 [TCP Dup ACK 131#1] 39956 → 80 [ACK] Seq=1 Ack=1 Win=432 Len…
257 16... 23.39.120.174     192.168.1.66      TCP      66 [TCP Dup ACK 132#1] [TCP ACKed unseen segment] 80 → 39956 [A…
258 17... 192.168.1.66      192.168.1.65      TCP      66 [TCP Dup ACK 152#1] 1714 → 46379 [ACK] Seq=1 Ack=1 Win=235 L…
259 17... 192.168.1.66      192.168.1.177     HTTP   1134 POST /jsonrpc.js HTTP/1.1  (application/x-www-form-urlencode…
260 17... 192.168.1.177     192.168.1.66      TCP      66 9001 → 43784 [ACK] Seq=1894 Ack=4336 Win=661 Len=0 TSval=990…
261 17... 192.168.1.177     192.168.1.66      HTTP    648 HTTP/1.1 200 OK  (application/json)
262 17... 192.168.1.66      192.168.1.177     TCP      66 43784 → 9001 [ACK] Seq=4336 Ack=2476 Win=1204 Len=0 TSval=13…
263 17... 192.168.1.66      213.138.101.172   TCP     110 36630 → 7779 [PSH, ACK] Seq=187 Ack=28035 Win=1444 Len=44 TS…
264 17... 213.138.101.172   192.168.1.66      TCP      66 7779 → 36630 [ACK] Seq=28035 Ack=231 Win=139 Len=0 TSval=187…
265 18... 213.138.101.172   192.168.1.66      TCP     204 7779 → 36630 [PSH, ACK] Seq=28035 Ack=231 Win=139 Len=138 TS…
266 18... 192.168.1.66      213.138.101.172   TCP      66 36630 →
267 18... 192.168.1.65      192.168.1.66      TCP      66 [TCP Dup
268 18... 192.168.1.66      166.78.143.191    TCP      66 [TCP Dup
269 18... 213.138.101.172   192.168.1.66      TCP     246 7779 → 3
270 18... 192.168.1.66      213.138.101.172   TCP      66 36630 →
271 18... 213.138.101.172   192.168.1.66      TCP     778 7779 → 3
272 18... 192.168.1.66      213.138.101.172   TCP      66 36630 →

▸ Frame 271: 778 bytes on wire (6224 bits), 778 bytes captured (6224
▸ Ethernet II, Src: Sagemcom_19:03:2a (40:f2:01:19:03:2a), Dst: Appl
▸ Internet Protocol Version 4, Src: 213.138.101.172, Dst: 192.168.1.
▸ Transmission Control Protocol, Src Port: 7779 (7779), Dst Port: 36
▸ Data (712 bytes)

0000  b8 e8 56 38 87 42 40 f2  01 19 03 2a 08 00 45 00   ..V8.B@.
0010  02 fc 93 92 40 00 36 06  b1 48 d5 8a 65 ac c0 a8   ....@.6.
0020  01 42 1e 63 8f 16 e5 5b  08 a5 cb 4d cf 46 80 18   .B.c...[
0030  00 8b bf 4b 00 00 01 01  08 0a 40 44 0f c1 00 ca   ...K....
0040  41 19 17 03 03 00 87 d5  1d b   04 a8 67 72 78   A.......
0050  00 8c 4d a8 fc ef 04 7f  d9 c   6b 22 4e 46 b0   ..M.....
0060  9e 05 46 2b 9e 0a f0 47  8b 99 b5 ca 98 24 5f 3e   ..F+...G
0070  f9 30 e5 06 f3 27 ca 57  8d ec 93 b4 d7 cd 08 16   .0...'.W
0080  cd 3e ce c7 30 fb a1 ff  6c 84 ed 4d 5d 7a 67 fb   .>..0...
```

Wireshark · Display Filter Expression

Field Name	Relation
http.response.code · Status Code	is present
http.response.line · Response line	==
http.response.phrase · Response Phrase	!=
http.sec_websocket_accept · Sec-WebSo…	>
http.sec_websocket_extensions · Sec-W…	<
http.sec_websocket_key · Sec-WebSock…	>=
http.sec_websocket_protocol · Sec-Web…	
http.sec_websocket_version · Sec-WebS…	
http.server · Server	
http.set_cookie · Set-Cookie	
http.ssl_port · Expert Info	
http.subdissector_failed · Expert Info	
http.time · Time since request	
http.transfer_encoding · Transfer-Encoding	

Value (character string)

Predefined Values

Range (offset:length)

Search:

http.set_cookie

Click OK to insert this filter

? Help ✓ OK × Cancel

1 **Packet Capture** *Wireshark* can work on a pre-recorded buffer of data or capture data live. 2 **Analysis** Use these menus to view statistics, run LUA scrips and see A–B conversations. 3 **Filter** *Wireshark*'s power comes from its ability to focus on the packets of most interest. 4 **Expression** Filtering is done with expressions, which can be built using the point-and-click list of protocols. 5 **Packet list** The timeframe, source and destination address of each packet. 6 **Protocol** Packets are coloured according to protocol. For example, green for HTTP; black for errors; violet for TCP. 7 **Packet details** Select a packet to see its metadata. 8 **Packet contents** Like a Hex editor, you can see the contents of a packet and even decrypt certain protocols.

Getting to grips with *Wireshark* has never been easy, but like amateur astronomy, the easiest way to start is to first take a look. On most systems, this that means that part of the installation needs root access to be able to listen to your computer's various network hardware. The recommended method is to isolate these privileges to **/usr/bin/dumpcap**, usually by adding your user to the **wireshark** group after installation.

Across the wire

Wireshark's power comes from the filtering mechanism, and while its incredible control does come from writing expressions to describe what you want to see, you can get close by selecting a packet and using its attributes to create a new filter, or by using the 'Expression' button. This opens a huge window full of protocol specifics and relationship filters so you can zone in on specific protocol packets.

It's complex but also wonderfully educational – a rare thing where you get to see network theory in action, like an electron microscope capable of real-time feedback from an Ethernet connection. Enter **http.content_type == "audio/ogg"** to see yourself download our latest podcast, for example, but you could also use the 'Find Packet' function to look for specific ASCII strings within a packet, all without knowing anything about networking.

PROJECT WEBSITE
https://www.wireshark.org

Oscilloscope

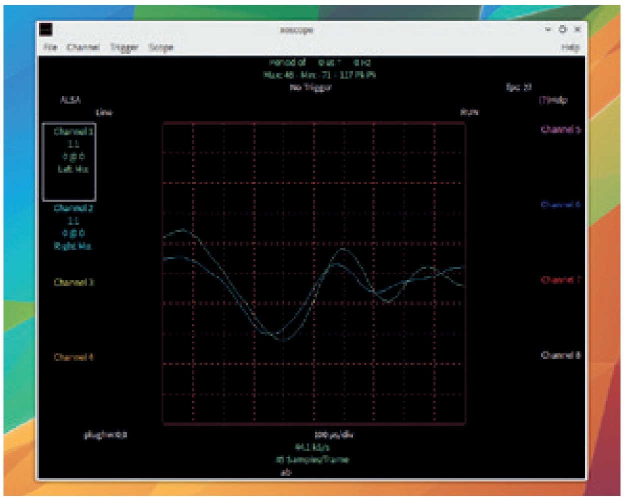
Xoscope 2.1

We have very fond memories from the late 80s of first getting hold of an audio sampler for the Amiga 500. We also had days of fun with tools like *Aegis Audiomaster II*, which visualised the shape of the audio being digitised, just as you view changes in voltage with an oscilloscope. Try to whistle a sine wave, for example, or see what happens when you whistle up or down an octave, or introduce other shapes and waveforms. It's immediate and addictive, and because these changes in audio are analogous to changes in voltage, it's teaching you about circuit design too.

Of course, you can still use an oscilloscope to visualise audio today, but it's not worth spending money on. Which is why *Xoscope* is so interesting. It connects to ALSA directly, as well as more professional sensors if you want to take it further, and it enables you to do all the things we used to do.

If you're very careful, you can even use it to monitor real changes in voltage on the audio inputs, but you can easily destroy your hardware if you monitor voltages above those tolerated by your inputs, so we'd recommend sticking with audio. As most distributions use *PulseAudio* for talking to their audio hardware, you may need to disable or pause it before launching *Xoscope*. We found that running **pasuspender -- xoscope -D ALSA** worked with our laptop hardware, but you may also need to specify which audio device to use with the **-A hw:0,0** argument. With audio entering an input or a microphone, you should be able to see the oscilloscope bounce along to the input and experiment with the sounds.

> You can even use Xoscope to monitor real changes in voltage on audio inputs


As well as audio inputs, *Xoscope* can use what are known as 'Comedi' sources for input data, such as analogue digital converters.

PROJECT WEBSITE
http://xoscope.sourceforge.net

File management

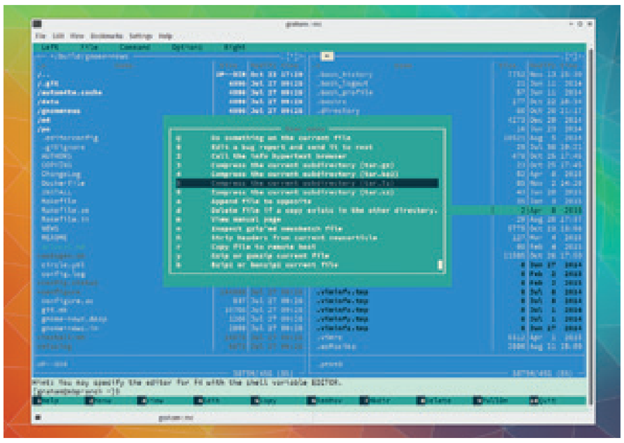
Midnight Commander 4.8.15

A generation of Unix geeks have relied on *Midnight Commander* to make file management on the command line a little more intuitive. It takes one of the best reasons for using a graphical desktop – a point and click interface in which files and folders can be visually manipulated – and transports the same functions to an easily navigable command-driven interface.

The idea first coalesced as the orthodox file manager in the early 90s, where this kind of functionality was a necessity on DOS machines without command histories and auto-completion. But when so many of us run low-powered machines, or low-end servers, its *Bash*-friendly interface and graphical style make it just as important today.

Over its long history, there have been various maintainers and developers, periods where there was little progress and periods where MC has flourished. It's currently flourishing, which is why we couldn't wait for a major release update to cover a new version in these pages.

One update follows another, and *Midnight Commander* is still going from strength to strength, from Samba support and Python APIs, to spellchecking, search and Zip, Rar, ARJ, and Tar handling. It's quick and requires very few resources, other than the libraries for all this added functionality. You can Tab between two locations and use the Ctrl+number menus to invoke functions. It's impossible to do **mc** justice here, but the best thing about it is that you already know how to use most of its functions, as it works just like any other file manager, just with the added advantage of running from the command line.


Whether you're using SSH or a Terminal running on your desktop, little can touch Midnight Commander for its power and flexibility.

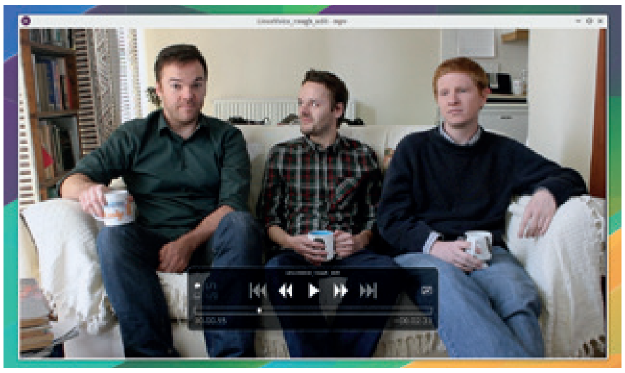
PROJECT WEBSITE
www.midnight-commander.org

Movie player

mpv v0.10.0

Mplayer has been around for a long time. Wikipedia says since the year 2000, but it feels longer to us. We remember using it alongside *Xv* to display images back when icons were chunky and colourful. All those years of development haven't made *Mplayer* any prettier, but it's still one of our favourite tools for no-fuss, rock-solid playback.

Mpv is a fork of the venerable *Mplayer*, a fork created to enable its developers to remove much of what they called *Mplayer*'s cruft, "features which stopped making sense 10 years ago", and by giving them the ability to add many modern features without having to worry about fitting in with the original project. The result is a brilliant movie player that focuses on the basics – playing moving media without worrying about offering every feature under the sun. Launch the application and drag a media file into the window. That's all there is to it. There are no menus and the transport controls only appear after you move the mouse into the lower section of the window or screen. This is because you interact with the player using the keyboard – cursor keys to seek, for instance, brackets for changing playback speed, and numbers for contrast, brightness, gamma and saturation. It's a distraction-free player with high-quality playback, which is exactly what you need. *Mpv* is quick and powerful and its keys can easily be remapped to a remote or used from an external keyboard. In common with its *Mplayer* roots, *mpv*'s command line interface is far more powerful than its simple playback window would suggest. Typing **mpv --list-options | wc** to count the number of lines of options in the latest version returned 454, which is a colossal number and far more than we can even abbreviate here. We highly recommend you take a look!


The company behind the *Plex* media player has just hired the main developer behind *mpv*, and is now basing its entire product around it rather than using the competing *Kodi* player.

Mpv is a distraction-free movie player with high-quality playback

PROJECT WEBSITE
http://mpv.io

Audio System

PulseAudio 7.1

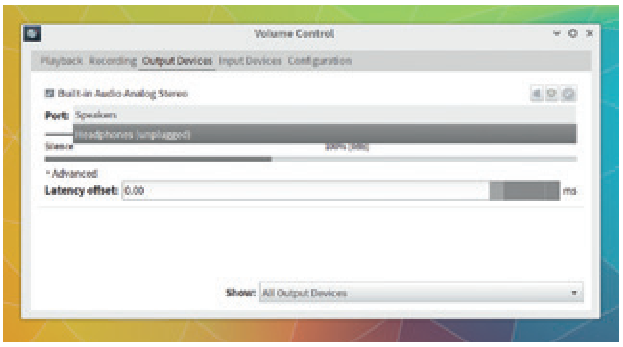
This is a major release of the audio subsystem used by nearly every Linux distribution, with 7.1 coming just weeks after 7.0 to fix a few bugs. This means that most users aren't going to notice any differences.

PulseAudio remains incredibly stable, and we've had much better luck with the new version talking to old ALSA applications or Jack. Most users don't want to mess around with virtual mixers, or edit init scripts, or close processes that are already using your audio hardware.

This is exactly what PulseAudio delivers the vast majority of the time, which is why so many distributions use it. But if it shares one thing with *Systemd*, its sister project also developed by Lennart Poettering, it's over-complexity. Even a normal user has to wade through *PulseAudio*'s brain numbing nomenclature.

A maze of configuration

The changes for each release are a good example of this complexity. The big addition for 7.0 was 'LFE channel synthesis with low-pass filtering', for example. It sounds complicated, but all that happens is that your bass speaker will now work properly, if you happen to use one. Rather than sending all the audio to a speaker that's only capable of playing back sounds below a certain low frequency, PulseAudio will now filter out the bits the speaker can't play and send only the bits it can. This will improve the audio quality on speakers that don't filter the sound themselves. There are lots of smaller changes too, including improved 'jack' detection. This initially got us excited, because PulseAudio doing more to help Jack would be a good thing. However, this isn't the right kind of Jack: *PulseAudio* is now more aware of hardware that can detect when speakers or headphones are connected to their 'jack' ports, allowing your audio playback to switch automatically.



Port: Speakers
Latency offset: 0.00 ms
Show: All Output Devices

Support for LFE channel synthesis means you'll finally hear the sound of a Death Star imploding.

PROJECT WEBSITE
www.freedesktop.org/wiki/Software/PulseAudio

Simple VPN

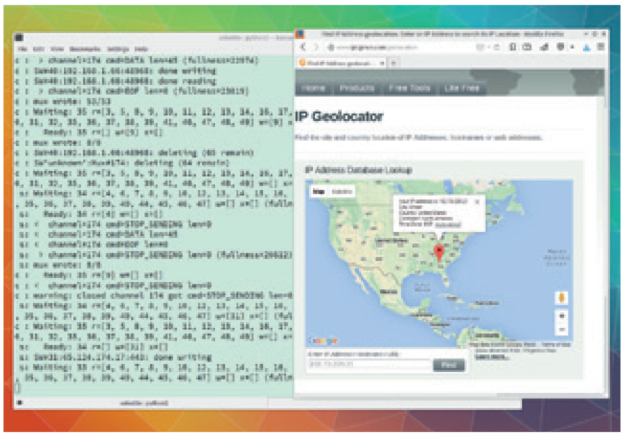
sshuttle 0.72

One of SSH's best feature is that you can use it as a simple SOCKS proxy server to tunnel network requests from one machine through the internet connection of another. This means, for example, you can access sites that are geographically restricted from location A by connecting to an SSH server in location B, where those sites are available. As your web requests and pages travel through SSH, they're also encrypted between locations A and B.

All you need to do to enable this magic is type **ssh -D 8080 user@locationb** from location A, changing the port number if this doesn't work. On your location A machine, you then need to either configure your connection to use location B as a SOCKS proxy, or use your browser's Advanced settings to configure SOCKS for web browsing.

But there are some limitations with SSH's SOCKS implementation. SOCKS isn't always supported, for instance, and you may need admin permissions at location B, which isn't always possible. Plus the performance of SOCKS through SSH isn't great. Which is where **sshuttle** comes in. It's a more functional VPN without any complications or dependencies – it doesn't even need to be installed on the server at location B, only an SSH account. On the client (location A), just make sure you've got Python installed and either install the **sshuttle** package if available or download the latest version from GitHub (**git clone git://github.com/apenwarr/sshuttle**). Now simply run the following command: **./'sshuttle --dns -vvr user@locationb 0/0'**. You'll be asked for your local password to get **sudo** access and the remote SSH user's password, if you use one, but after that your VPN is up and running without any further configuration.

sshuttle is a more functional VPN with no complications or dependendencies



Don't tell anyone, but there's a rumour that sshuttle's DNS routing can bypass geographical restrictions in services like Netflix.

PROJECT WEBSITE
https://github.com/apenwarr/sshuttle

Note taking app

Cherrytree 0.35.11

Applications that help you make and manage your own notes are a little like word processors. They're incredibly useful, and can easily become an essential part of your workflow, but they're difficult to get excited about and require discipline to use effectively.

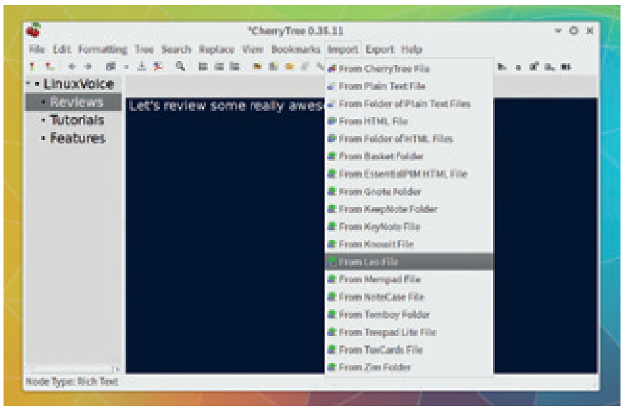
But they're genuinely worth the effort, especially when you're working on large or complex projects. Students, for example, often make hundreds of different notes across many different subjects and durations. For those notes to mean anything, they need to have context and be useful during revision, and the same principle applies to large projects or even putting a magazine together. *Cherrytree* is a note-taking application that attempts to make adding this context as easy and as transparent as possible, while giving you lots of control over how your notes look and are retrieved.

Order, order!

Like the filesystem on our computers, your notes are organised in a hierarchical tree of nodes. You might create one for a specific magazine issue, for example, or for college years and then a course.

Notes can be entered into the text editor for both the nodes that act as folders and for the nodes themselves. The text editor has excellent syntax highlighting, list and image embedding, and HTML-styled markup. There's also plenty of control over how the application appears – you can remove the muticoloured cherries used to represent nodes, for example, and replace them with simple bullets, and you have complete control over colours and font sizes.

You can choose to export a specific node or sub-nodes, for instance, and output formats include HTML, text, PDF and as an HTML-styled table of contents.



Cherrytree also supports a huge number of import formats for bringing in your collection of notes.

PROJECT WEBSITE
http://www.giuspen.com/cherrytree

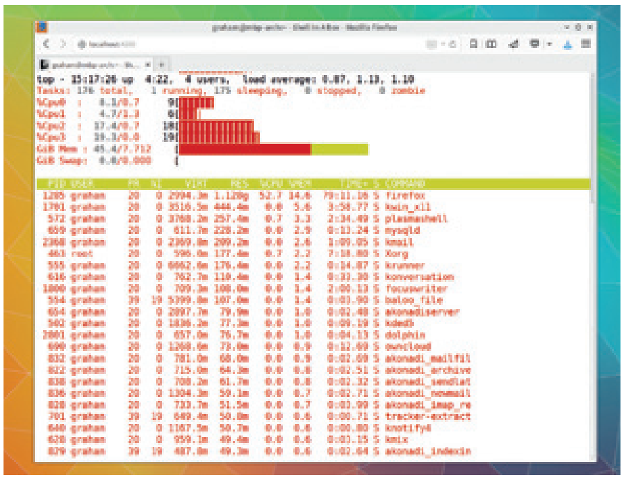
Remote shell access

Shellinabox (unofficial fork)

Initially, the thought of being able to execute shell commands from a web interface may seem like a bad idea. Most of us would immediately think of the security implications. But it's also a very cool idea and there's a useful niche for something like this, especially when placed on any number of devices running in your home or behind any firewall.

At its best, *Shellinabox* is really a glorified SSH client, but with one important difference. Rather than requiring a shell or a terminal to access all the loveliness of SSH, all you need is a web browser and *Shellinabox* running somewhere you can access. Installation onto that 'somewhere' is very straightforward. Grab and install the package and most distributions will install an init script to launch the daemon at boot. You may want to edit its default settings from **/etc/default/shellinabox** or **/etc/sysconfig**, especially as you can limit access by IP address. This is the best option, as the daemon will have the correct privileges to access local and remote servers – you can even use a local instance to connect to any remote SSH server. You can provide local access to your own account with **shellinaboxd -t -s /:AUTH:HOME:/bin/bash**.

With the daemon running, you can now point a web browser at your machine's IP address, or **https://localhost:4200** for local access. 4200 is the default port unless you change this from the configuration file, and *Shellinabox* does run through SSL so that data between your browser and the server is encrypted with a self-signed certificate. You'll need to add an exception to this in *Firefox*, for example, if it complains too loudly, or SSL can be disabled by adding the **-t** argument when you launch the daemon.



```
top - 15:17:26 up  4:22,  4 users,  load average: 0.87, 1.13, 1.10
Tasks: 176 total,   1 running, 175 sleeping,   0 stopped,   0 zombie
%Cpu0  :   8.1/0.7     9[
%Cpu1  :   4.7/1.3     6[
%Cpu2  :  17.4/0.7    18[
%Cpu3  :  19.3/0.0    19[
GiB Mem : 45.4/7.712    [
GiB Swap:  0.0/0.000    [

  PID USER      PR  NI    VIRT    RES  %CPU %MEM     TIME+ S COMMAND
 1285 graham    20   0 2994.3m 1.120g  52.7 14.6  79:11.16 S firefox
 1701 graham    20   0 3516.5m 444.4m   0.0  5.6   3:58.77 S kwin_x11
  572 graham    20   0 3768.2m 257.4m   0.7  3.3   2:34.49 S plasmashell
  659 graham    20   0  611.7m 228.2m   0.0  2.9   0:13.24 S mysqld
 2368 graham    20   0 2369.0m 209.2m   0.0  2.6   1:09.05 S kmail
  463 root      20   0  596.0m 177.4m   0.7  2.2   7:18.80 S Xorg
  555 graham    20   0 6662.6m 176.4m   0.0  2.2   0:14.87 S krunner
  616 graham    20   0  762.7m 110.4m   0.0  1.4   0:33.30 S konversation
 1800 graham    20   0  709.3m 108.0m   0.0  1.4   2:00.13 S focuswriter
  554 graham    39  19 5399.0m 107.0m   0.0  1.4   0:03.90 S baloo_file
  604 graham    20   0 2097.7m  79.9m   0.0  1.0   0:02.48 S akonadiserver
  502 graham    20   0 1836.2m  77.3m   0.0  1.0   0:09.19 S kdeds
 2001 graham    20   0  657.0m  76.7m   0.0  1.0   0:04.13 S dolphin
  690 graham    20   0 1268.6m  73.0m   0.0  0.9   0:12.69 S owncloud
  832 graham    20   0  781.0m  68.0m   0.0  0.9   0:02.69 S akonadi_mailfil
  822 graham    20   0  715.0m  64.3m   0.0  0.8   0:02.51 S akonadi_archive
  838 graham    20   0  708.2m  61.7m   0.0  0.8   0:02.32 S akonadi_sendlat
  836 graham    20   0 1304.3m  59.1m   0.0  0.7   0:02.71 S akonadi_newmail
  828 graham    20   0  733.7m  51.5m   0.0  0.7   0:03.99 S akonadi_imap_re
  701 graham    39  19  649.4m  50.0m   0.0  0.6   0:00.71 S tracker-extract
  640 graham    20   0 1167.5m  50.7m   0.0  0.6   0:00.80 S knotify4
  628 graham    20   0  959.1m  49.4m   0.0  0.6   0:03.15 S kmix
  829 graham    39  19  487.0m  49.3m   0.0  0.6   0:02.64 S akonadi_indexin
```

Install *Shellinabox* and get lovely terminal goodness from anything with a web browser, from games consoles to mobile phones.

PROJECT WEBSITE
https://github.com/shellinabox/shellinabox

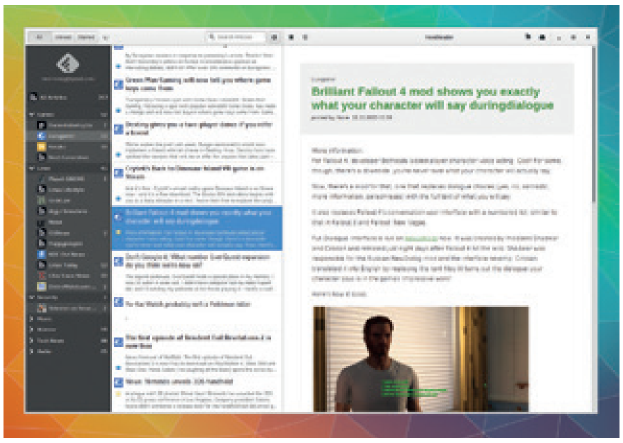
RSS reader

FeedReader 1.4

Like many people, we relied on Google's now-defunct cloud RSS feed aggregator. It enabled you to easily accrue automatic updates for sites on subjects you cared about, and its simple categorisation and hierarchical organisation meant you could easily keep abreast of 1000s of updates a day. For anyone working in the field of technology, it was the only way of sifting the inconsequential from the important.

With the possible exception of *Feadly* and some of the clients that can access its API, the closest alternative we've found to Google Reader is the News app for OwnCloud. This enables you to add RSS aggregation and browsing from a browser session connected to your own OwnCloud account. It looks very similar to Google's product and has the advantage of being completely private to your OwnCloud account and installation.

There are also a considerable number of standalone applications, but once you've got used to cloud synchronisation across multiple devices, it's difficult to go back to an offline RSS reader. Which is why *FeedReader* is so good. It connects to an online service and allows you to browse your feeds, feeling very similar to *Google Reader*. It works exactly how you'd expect it do, and the HTML rendering is excellent. Feedly is supported and works well, but more importantly, the latest release also adds support for OwnCloud, letting you connect the client to your own server with the News app installed and access your RSS feeds just as you would from a third-party site like *Feedly*.


Brilliant Fallout 4 mod shows you exactly what your character will say duringdialogue

RSS aggregation is still alive and well thanks to great applications like FeedReader

It's a great alternative to the web interface and is even preferable for us, despite OwnCloud's News app improving solidly since its release close to the end of *Google Reader*. *FeedReader* is also a fully fledged Gnome application and takes advantage of the new design minimalism, which we like.

PROJECT WEBSITE
https://jangernert.github.io/feedreader

FOSSPICKS Brain Relaxers

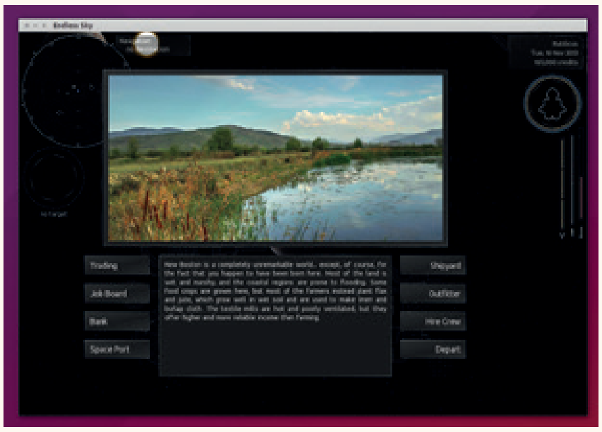
Space trading

Endless Sky

Linux may not (yet!?) have David Braben's *Elite*/ *Frontier* sequel, *Elite Dangerous*, but it's got a few great alternatives for lovers of space trading and strategy. *Endless Sky* is the latest we've found, and while it's available on Steam, it's also open source and hosted on GitHub, and available as binary packages for all the popular distributions. The only slight hitch is that it requires OpenGL version 3.0 or higher, which means you'll need a graphics card capable of more than legacy drivers.

The game itself isn't visually demanding, relying instead on static text and illustrations and a real time *Asteroids* top-down view of your spaceship as it travels from one system to another. It still looks great, and the planets are all accurately pre-rendered and lit according to the position of the sun, although distances and alignment are complete fantasy. Like *Asteroids*, your ship has its own inertia that you need to control while trying to target other craft or land on a new planet.

You pop into and out of the background as you find places to land, as do other craft controlled by AI. The game mechanic relies on trading, completing missions and exploring, although combat is never far away. The missions themselves usually involve delivering something from one place to another for a payment, and with the money you earn you can upgrade your ship and hire people to help you. *Endless Sky* reminds us of a text-based game for Palm OS called *Space Trader*, where you shuffle through menus buying inventory and travelling around the system, and we'd suggest it may be ideal for younger people as they can learn a lot about buying, selling and trying to make money while having a great deal of fun in a wonderful environment.



Imagine what would happen if you could take your ship out of the arcade classic *Asteroids*, and use it to fly, trade and fight – that's *Endless Sky*.

PROJECT WEBSITE
https://endless-sky.github.io

Real time strategy

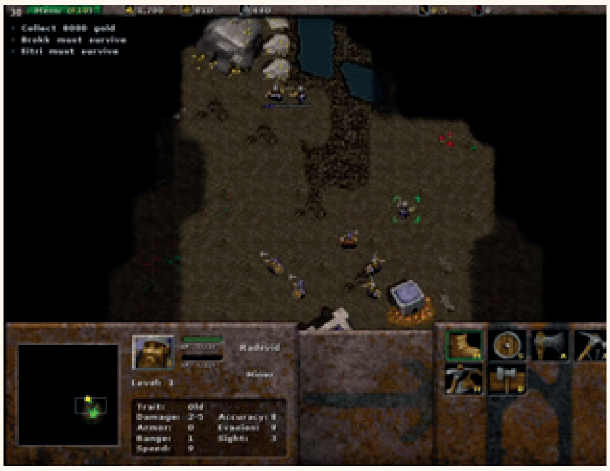
Wyrmsun 1.7.0

It's widely accepted that the idea of harvesting limited resources to build your defences and attack some foe across a hidden map started with the 1992 game *Dune II*. That game was incredibly addictive, but it's still surprising that something relatively complex could lead to such a popular genre, and one that's still spawning new games today – such as *Wyrmsun*.

What makes *Wyrmsun* different is that it's both a commercial game on Steam (currently priced at £3.99) and an open source/ GPL game in its own right. The commercial aspect in particular means the game is already at a high standard of both gameplay and graphics, even if you're building the open source version, and the scenarios, characters and maps are also particularly well thought out.

The graphics themselves stay true to the retro feel of the genre, with a top-down view that will feel familiar to *Rogue* players. Like *Rogue*, many of the quests are constrained to corridors and rooms within buildings, with a fog of war lifting as you walk to reveal more details on the map. This is different to the open world of most similar games and reminds us more of *Warcraft* than the original real-time strategy games. It means you pull your comrades through each level as an adventurer rather than a strategist.

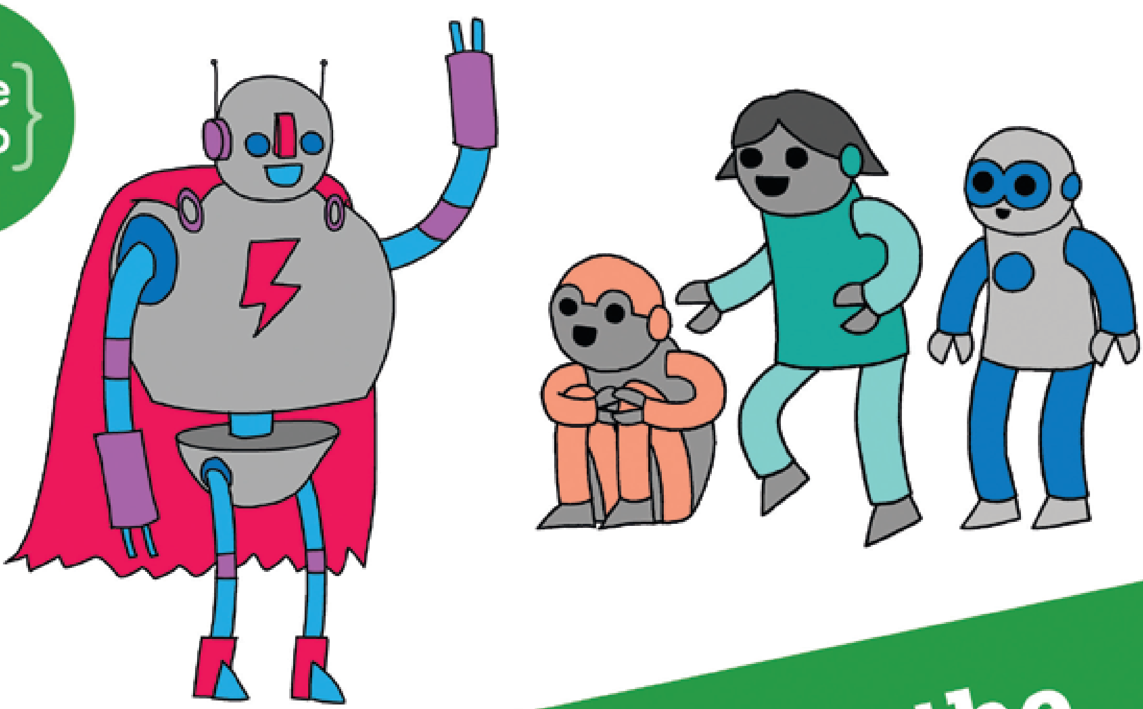
There are quests for different races, with technology trees and skills to unlock, all of which can be used as upgrades within the game. For the price, it's brilliant, and considering it's also open source, highly recommended. LV



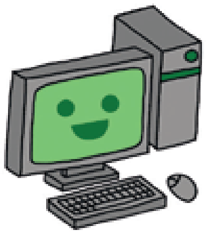
Open source takes on Steam with a great real time strategy game that's available to buy and to tinker with.

PROJECT WEBSITE
https://github.com/andrettin/wyrmsun

{code club}



Can you help inspire the next generation of coders?



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

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

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

There are loads of ways to get involved!

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

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
is never happier than when he's prying open a box and reprogramming the gubbins inside.

This month I've been working with a lot of small embedded devices, both in the smart homes feature and in the tutorial on reusing old Android phones. Yet again, I've been reminded of the importance of hackable devices. Most embedded devices are closed off, and don't allow the user any option to meddle and change their function. This might make life a little easier for the manufacturer, but they make things much worse for users.

The more control you have over a particular piece of hardware, the more uses you can put it too. In cases like a smart home, this gives you more control of how you set up something as important as your home environment. In the case of an Android phone, this gives you the ability to repurpose your hardware once you no longer need it for its intended use. In all cases, it empowers you to be a user of technology rather than a consumer of disposable devices, and that's what we should all be striving for.

The more open a device is, the more possibilities it has to make the world a better place, and this should be the aim of all technology.
ben@linuxvoice.com

ANDROID: REPURPOSE YOUR OLD PHONE

Android devices are more than just phones and can live on in other guises.

BEN EVERARD

WHY DO THIS?
- Reduce electronic waste and save the environment
- Make better use of the hardware you already own
- Improve the security, look or connectivity of your home

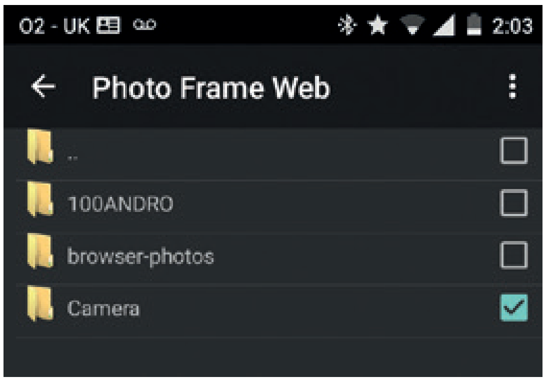
How often do you replace your phone? For some people it's every year, but some people hang on until the device is completely dead before upgrading (every two years is about average, give or take six months). Whichever category you fall into, there's a pretty good chance that you've got one or more old Android phones lying around the place no longer being used. Even if you stubbornly refuse to replace a working phone, there's a good chance that you've got one with a dead and unreplaceable battery that still works when the phone's plugged in.

This mountain of unused phones is an untapped resource, and it would truly be a shame if they were left to rot. Instead, we're going to take a look at various ways that you can still use your phone hardware even when you no longer need it as a phone. After all, they're still little computers with a Wi-Fi connection, a camera and possibly a battery.

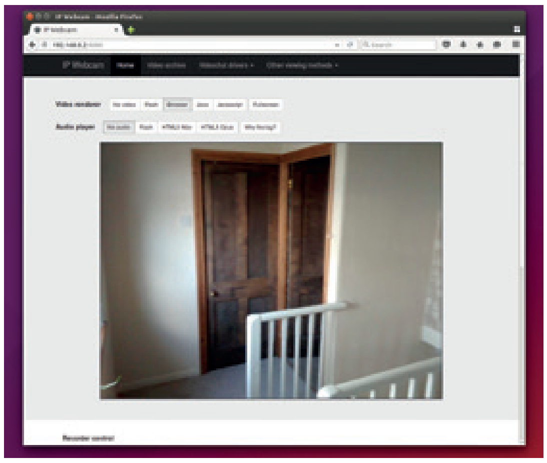
> A phone is just a little computer with a Wi-Fi connection, a camera and possibly a battery

You may find that your phone refuses to work without a sim card in it. If so, you'll need to purchase a pay-as-you-go card and just put the minimum credit on it.

Perhaps the most obvious thing to do with a small computer with a camera and a network connection is turn it into a webcam. You could use this for anything from security to a baby monitor to watching nature (such as in a birdbox). You don't need a good screen or battery for this (as you can leave it plugged in). As long as the camera still works and the phone still boots, you can convert it into a webcam.



Re-live your glory days by watching old photos on the old phone that you used to take them.



Thieves beware: Linux Voice towers is protected by a network of old phones.

There are quite a few apps for Android that will convert your phone into a networked webcam, the best of which (for our purposes) is *IP Webcam* by Pavel Khlebovich. All you have to do is install the app, then open it and press Start Server. You'll see the URL of your web camera on the screen, so visit this on any other machine connected to the same Wi-Fi network. On that webpage, you can create a video stream (just click on one of the options for video to enable the image). You can also schedule video capture, create rolling captures and more. If the stream has too few frames per second, you can downgrade the video quality using the on-screen slider.

Getting the picture

Like many people, we find ourselves taking lots of pictures, but hardly ever spend the time to look back through them. This isn't like the good old days when we took real pictures, the type that came back on paper, the type you could put in a frame on your wall or desk. If you've got an old Android phone with a working screen, you can relive those days by using it as a digital picture frame. Even if the battery's dead, you can keep it plugged in.

As with the webcam, there are a bewildering array of apps in the Google Play store to add this functionality. Almost all of them will make a slideshow out of the files you've got saved locally, and the better ones will also be able to pull images from online sources such as Facebook, Dropbox or Google Drive. It's just a matter of finding one that has the features you need and that isn't overloaded with adverts. We found that *Photo Frame Web* by Jeroen Wyseur worked well. After you've installed the app, you select the source of the images and start the slideshow. For an even more nostalgic experience, mount the old phone inside a picture frame.

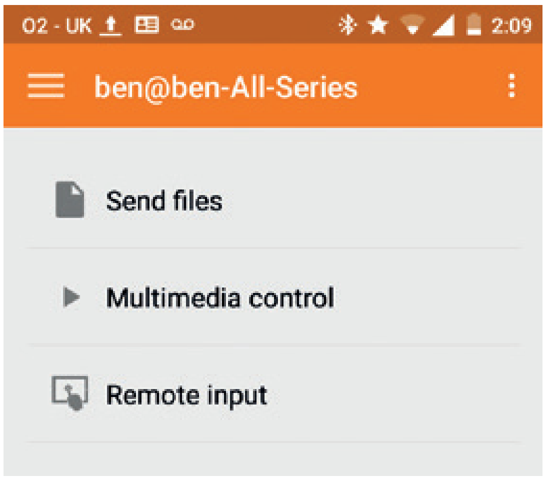
Picture frames aren't the only way you can re-purpose your phone as a internet-connected screen. You can use your old phone as a clock using any one of the hundreds of clock apps. As with the photo frame, the key here is finding a good-looking way to mount the phone so it doesn't look like an old phone lying around, but a stylish clock.

Seeking input

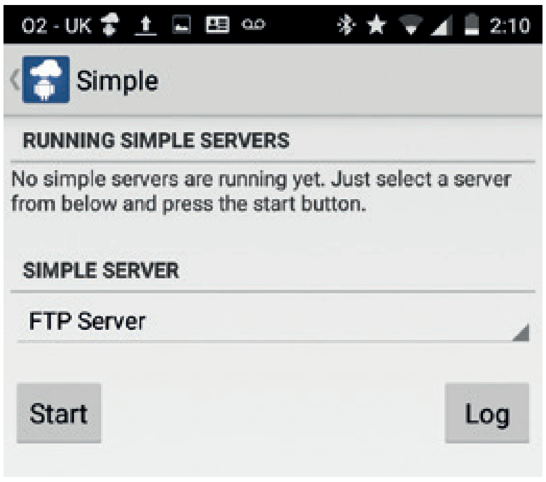
You can use an old phone as an input device for your Linux computer using *KDE Connect*. Despite the name, this should work on most desktop environments (although you will need to install the KDE framework in order to use it). You'll need the *KDE Connect* software installed on both your phone (from the Google Play store) and computer.

Once the software's installed, open the app on your phone, then on your Linux machine use the command **kdeconnect-cli -l** to find the devices. All devices have an associated ID that you'll need in order to connect, and this is the alphanumeric string that should be listed by the above command. You can then use the **--pair** option to connect the two devices together. With our ID, this was done with:

```
kdeconnect-cli --pair -d 3f01ca6d603c9c5b
```



As well as an input device, *KDE Connect* can be used to share files and notifications between your computer and your phone.



No simple servers are running yet. Just select a server from below and press the start button.

Servers Ultimate: for when you want to turn a drawer full of old phones into a data centre.

Now that the devices are paired, you can select the remote control option on your phone, and then the phone acts as a touchpad for your computer. If you need to enter text, you can hit the keyboard icon to bring up the onscreen keyboard.

Serve yourself

Phones are just small computers that are optimised to use power efficiently, and this makes them perfect for home servers. If there's any life left in the battery, they even have an uninterruptable power supply to keep your mini server humming even if mains power goes down.

KWS is a simple and efficient web server for Android. Just install it and hit Start Server to begin. There are a couple of useful options, including enabling directory listing and setting the home directory, but otherwise the defaults should be fine. Performance on our test device (a Moto G) was reasonable for text and HTML. Images and other larger files were noticeably slower than the network speed, but acceptable for light use.

Probably the most powerful server app for Android is *Servers Ultimate* by Ice Cold Apps. This enables you to run almost any type of server you can think of, including common options such as HTTP and FTP to more specialised options such as *Git* and IRC. The chances are that if you need a server, this app will be able to provide it on Android. You'll need a rooted phone if you want to be able to run the server on a normal port, but non-rooted phones can run most servers on the higher port numbers if you wish. *Servers Ultimate* is free for seven days, but if you want to keep using it, you'll need the Pro version for £7.50 – we think that's plenty of time to have a play with it and make an informed decision.

Ben Everard plays with hardware for fun and is the co-author of the excellent *Learning Python with Raspberry Pi*.

ARDOUR: SYNTH WORKSHOP

Upgrade your hipster credentials with some bleeps, plops, pings and squelches.

GRAHAM MORRISON

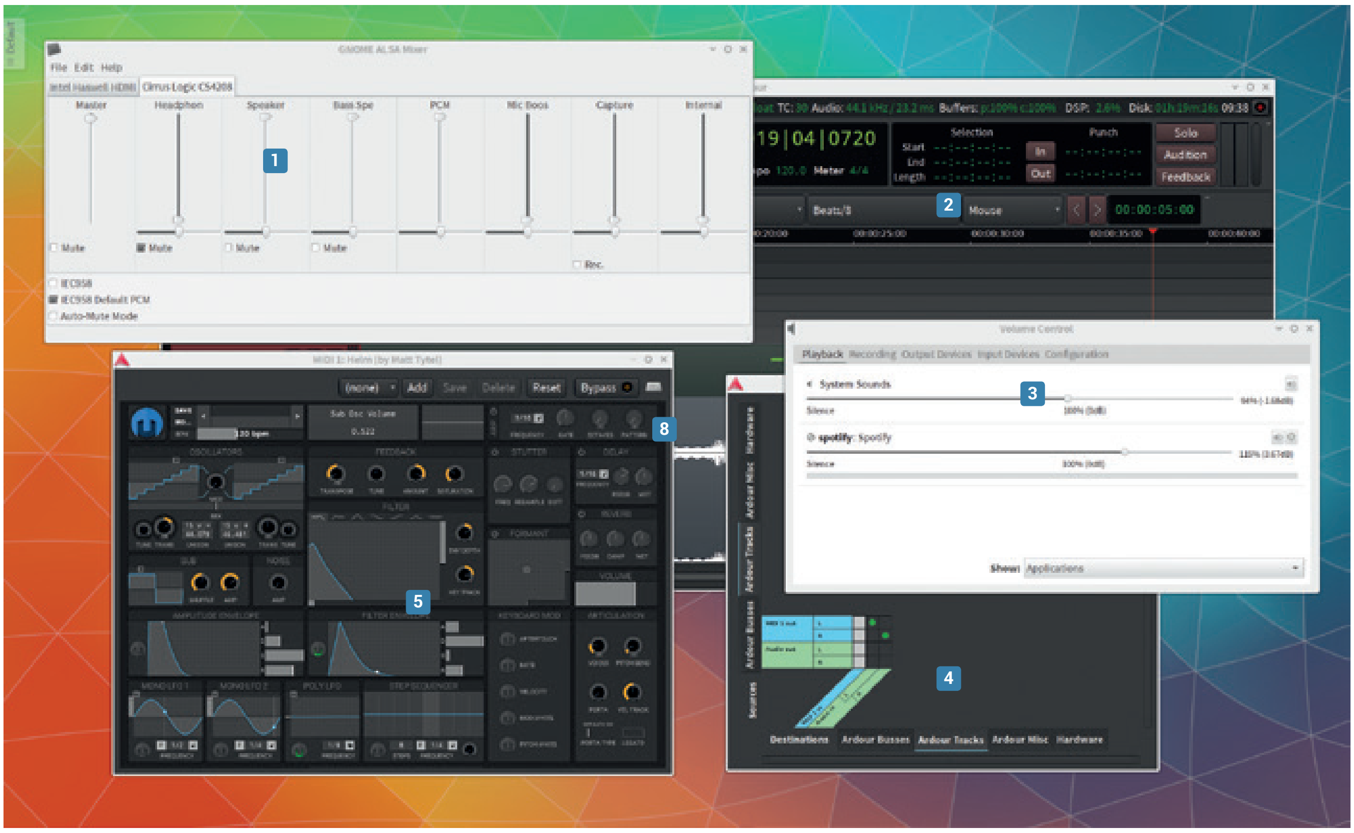
WHY DO THIS?

- Create awesome sounds…
- … without knowing anything about sounds…
- … then record a masterpiece

Software synthesizers,and the other lovely-sounding open source sound generators, are fully equivalent to the hardware devices that typically cost hundreds and thousands of your local currency. They're playable, tweakable, configurable and powerful, and can be used as the central parts of many different kinds of music. But synthesizers are also a great deal of fun. After a decade in the geeky wilderness, they've started to become popular again – forged into DIY cases, or wired into the storage of multi-buttoned Raspberry Pis. They're instant, educational and entertaining. It's one of the reasons we picked Sonic Pi as our educational tool of choice in a Group Test of software that teaches people to code.

But we've also had a couple of emails that asked what should be an easy question to answer: "How do you actually get these software instruments to work?" In formulating an answer, we quickly realised that it wasn't easy at all; there's a lot of assumed knowledge, big pieces of software to conquer and configurations to play around with. Almost the complete opposite to the switch-on-and-have-fun attitude we were describing. Which is why we're going to tackle the problem here by building an easy-to-use music/audio platform that will let you quickly play with synthesizers and audio effects without needing any prior knowledge of how it works, or having any specialist audio hardware.

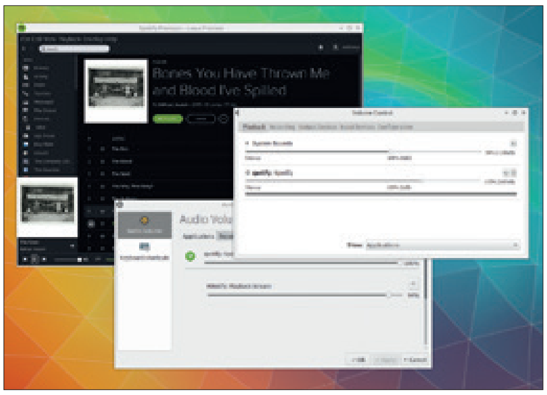
Overview of Linux Audio



1 ALSA Even when using other audio layers, ALSA is still responsible for talking to your audio hardware, and you can usually get ultimate control over your input and output levels by installing the *GTK Alsamixer*.
2 Ardour We've used Ardour as the plugin host. It's complex, like *Blender*, but powerful. If you want something simpler, *Qtractor* offers similar features.
3 PulseAudio Latest versions of *PulseAudio* will automatically pause when something else takes control of ALSA, such as Ardour or Jack, making it much easier to use with other audio software.
4 Jack Like a mixer in a recording studio, if you run 'Jack' you'll be able to connect any Jack-supported audio destination with any source. It's how Ardour works in the background.
5 Software synthesizer Many synths, like *Helm* here, will run as standalone applications, which is great for a quick play. But to record anything, they need to be paired with a host application.

1 What is Linux audio?

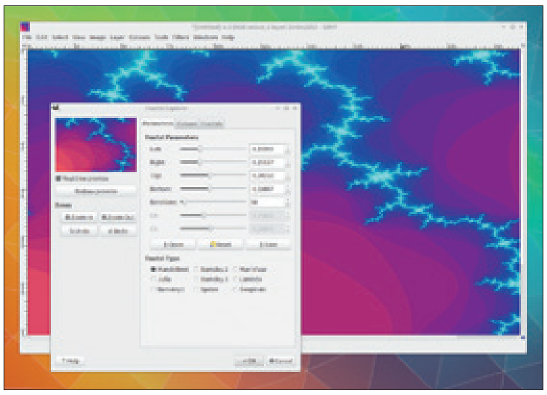
The days of hunting down a wayward Macromedia Flash process just to get *XMMS* working are gone. This is mostly thanks to *PulseAudio*, the overarching audio system used by the majority of distributions. *PulseAudio* straddles the complete audio Linux audio stack, from the hardware up to the desktop. Before *PulseAudio*, ALSA was used to drive the hardware (and ALSA's hardware drivers are still used by the kernel). ALSA would also speak to some applications, while at the same time, higher level sub-systems like artsD, Phonon, GStreamer or Xine would also battle for an application's attention. The result was usually silence. *PulseAudio* has brushed these aside, and with the sole exception of JACK, is now the only common Linux audio tool that talks to your hardware.



2 What are plugins?

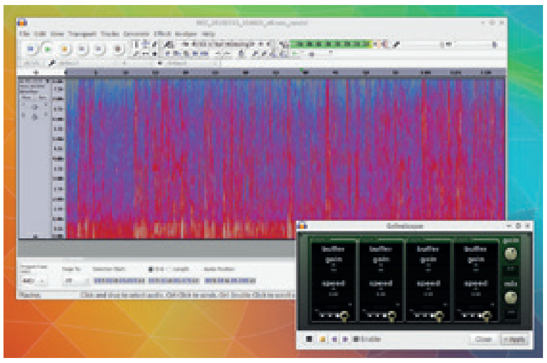
Most software synthesizers don't run as applications. Instead, they're installed as libraries and appear in host applications that support those libraries. This is also true for audio effects. Synthesizers and effects are called plugins because you plug them into your existing environment (in the world of the recording studio, you really do plug these things in with cables).

Plugins are analogous to filters and processes you use in graphics tools like *Gimp*, with a synthesizer perhaps being more like a fractal generator, and an echo effect a little like a drop-shadow. The important point is that even after you've installed a plugin, you won't be able to use it without launching a host application that knows how to find and open the plugin you've just installed.



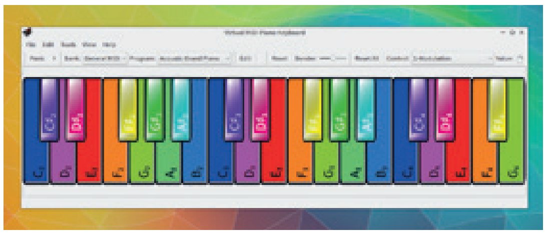
3 Libraries

There are lots of different plugin libraries for adding synthesizers and effects, but there only three modern formats that you're likely to come across. The first is called 'LV2'. This is the sequel to 'LADSPA', and it's the closest thing to a native open source format we have. The second common format is DSSI. This is used for synthesizers because they don't specifically process incoming audio, like an effect, but they respond to note information entering via MIDI or sometimes OSC, which is more advanced protocol for sending control data. Finally, there's the VST format. VST (Virtual Studio Technology) was developed by Steinberg in the 1990s, and is by far the dominant plugin format for Windows. Linux users can run Windows VST plugins under *Wine*, or more recently, native VST plugins built against a Linux version of the VST library.



4 MIDI

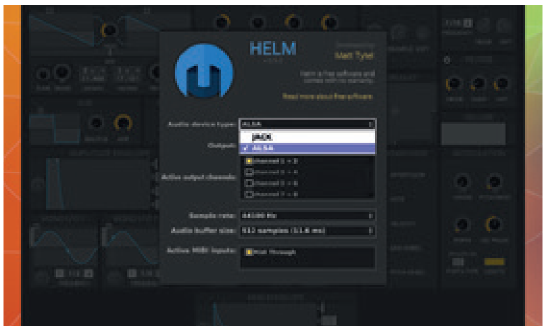
The final piece in the puzzle of Linux audio is MIDI support. MIDI is an ancient protocol that was used to program synthesizers in the 1980s – think Vince Clarke, long black cables and DIN sockets. MIDI encapsulates musical elements such as playing a note, the speed that the key or note is hit (velocity) and any changes that occur while sounding the note, such as the pressure you apply to a key or sliders you move. Despite being designed for hardware, MIDI still rules the software world too, with the protocol being central to how you interact with desktop synthesisers. ALSA handles MIDI, and you don't need a physical MIDI keyboard to generate MIDI data. The easiest way is to install a virtual MIDI keyboard such as *VMPK*. This sits on your desktop and turns mouse clicks and Qwerty presses into MIDI note data. It's the quickest and easiest way to get up and running. If you ever get Jack running, we'd also recommend *jack-keyboard*, as it does the same thing without you having to worry about MIDI devices: Jack handles everything.



5 Introducing Helm

We're going to start with our favourite synth: *Helm*. It sounds fantastic, but the best thing about *Helm* is that it just works. After you've installed it (we used version 0.50), you can launch it without worrying about MIDI or plugin hosts.

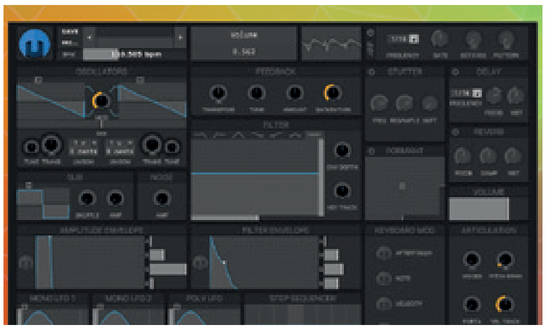
By default, it will plug itself into ALSA connected to *PulseAudio*, and respond to keypresses on your computer's keyboard by playing a note – just make sure ALSA is the selected audio device type in the preferences panel that appears when you click on *Helm*'s logo. We'd first recommend familiarising yourself with the various sounds that *Helm* creates, which can be done using the preset browser. Open it by clicking on the preset name just to the right of the logo in the top-left. It opens a simple patch navigator, and when you select a new patch you can play it with your keyboard.



6 Create your own sound

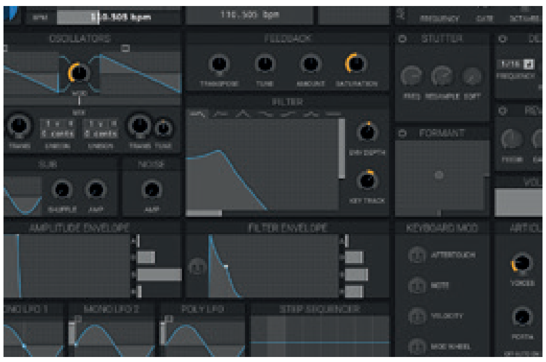
None of the preset sounds in *Helm* are good at teaching you how to create your own, so we're going to start from scratch. If you're running *Helm*, you'll need to restart it to get the sound to an initial state, and you may need to restart from the command line. *Helm*'s initial patch is a sine wave.

You can see this when you play in the oscilloscope, and you can see the two oscillators that make up this sound in the top-left. Drag the small square to the right to change these to 'Saw Up'. This is much more suitable, because it contains harmonics we can work on with other parts of the synth. To fatten the sound, turn the 'mod' knob between the oscillators to the mid point; this will mix the output from both equally, and slightly adjust the 'Tune' knob beneath one of the oscillators. This will add a phased effect as the waveforms of each oscillator shift in and out of phase.



7 Enabling polyphony

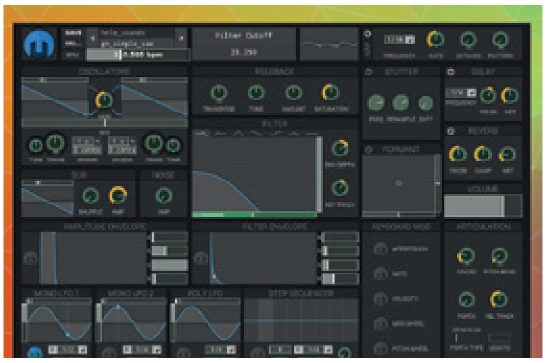
Helm's default state is monophonic. This is great for bass and lead sounds, but not so good if you want to play chords. Look for 'Voices' in the 'Articulation' section, bottom-right, and change the Voices value to 5 or more. Because of the way the two oscillators are now driving in and out of phase for every note you're playing, you'll get a rich string-like sound. We'll now filter out the high-harmonics, a process responsible for much of a synthesizer's character. The filter section is in the middle of the window, and you need to select the waveform on the far-left. Now drag the waveform to the left and up slightly to add a resonant edge at the cut-off point. We're now going to modulate the cut-off point with the value of LFO 1. Do this by clicking on LFO 1's *Helm* icon and click+dragging the horizontal slider beneath the filter waveform.



8 Envelopes

We're now going to use the Filter Envelope to change the cut-off frequency of the filter over time. First ramp up Env Depth a little in the Filter section (this governs how much of the filter will be affected), then adjust the ADSR values in the Filter Envelope. ADSR stands for Attack (key down), Decay (the length of time it takes to go from Attack to Sustain), Sustain (the level while the key is being held) and Release (the time it takes for the value to fade), and we've gone for A = 0; D = 0.2s; S=0.15; R=1.5.

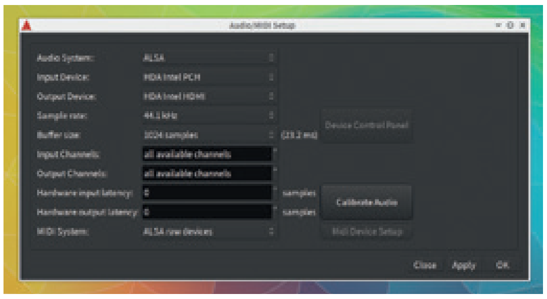
Finally, to add a lovely sparkle to your sound, enable the reverb section. This will add a naturalistic acoustic echo. You may also want to experiment with the delay section, or add a sub-bass element with the Sub oscillator, although you should choose a Saw- or Square-based waveform to achieve the best effect.



9 Getting sound somewhere else

We now need to start to consider recording the output, or at least launching a synthesizer in an environment where other sounds can be added to. This is where we need to run an audio host, and there's a quite a selection to choose from. At one end of the scale, there's Ardour, which is perhaps the audio equivalent of *Blender*. It's complicated and has a steep learning curve, but it's capable of professional results. At the other end of the scale there's something like *Radium*, which is an old-style tracker with support for all the best plugin and synth formats (including VST).

We're going to use Ardour, simply because we're not afraid of its power (and because it no longer needs Jack). Just install and launch. Select a New Session and make sure ALSA is selected as the audio subsystem. This should work even if you have *PulseAudio* installed and running, but you can also try Jack if you like, as this is run automatically too.



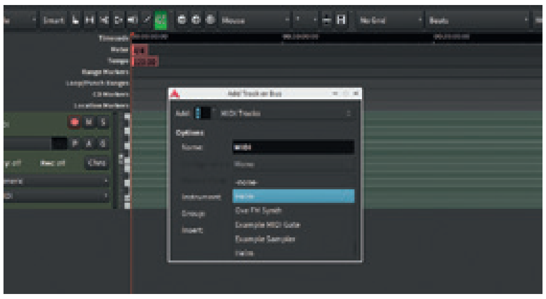
Audio/MIDI Setup

Audio System: ALSA
Input Device: HDA Intel PCH
Output Device: HDA Intel HDMI
Sample rate: 44.1 kHz
Buffer size: 1024 samples (23.2 ms)
Input Channels: all available channels
Output Channels: all available channels
Hardware input latency: 0 samples
Hardware output latency: 0 samples
MIDI System: ALSA raw devices

10 Creating the tracks

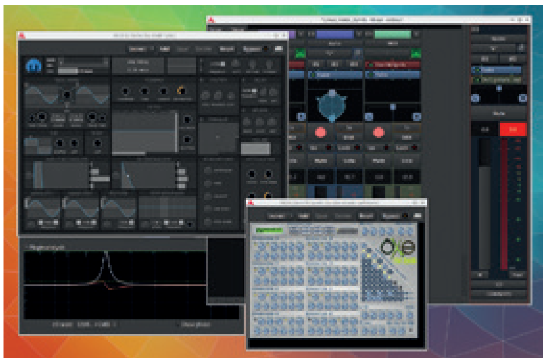
At a minimum, we need to create two tracks – one for handling the MIDI note data, and one for handling the audio that comes out of the synthesizer. You can do this by right-clicking the background of Ardour's main window. Create a MIDI track first and you should be able to choose *Helm*, or any other DSSI synth you have installed as the instrument. This will load the instrument as the destination for this MIDI track, and any other MIDI data coming into the application.

Now create an audio track. *Helm*'s output is stereo, so you should choose that. Ardour will automatically link the audio coming out of *Helm* into the input of the new track you've created. Generate some MIDI input or vertically expand the MIDI track and click on the notes to check. If no sound is generated, open Window > Audio Connections and make sure Audio In 1 & 2 for your audio track are checked against MIDI 1 L & R for the MIDI track.



11 Add instruments and effects

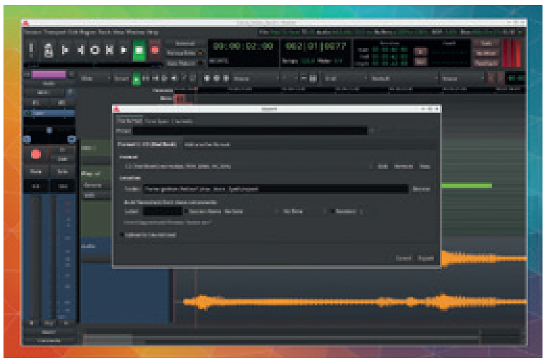
You can open the synth's control panel from either the Mixer or by enabling View > Show Editor Mixer. This will show the channel strip for the selected track, and when you select the MIDI track you'll see that *Helm* is listed as an insert at the top, above Fader. Double-click on the word to open the editor. You can add different effects or synthesizers to this insert point, although we'd recommend creating separate tracks for separate audio generators. We added a new track for the *Oxe* synthesizer, for instance. Both outputs will be mixed together, which you can see on the master channel of the mixer. The mixer view, Window > Mixer, is perfect for changing the volume of each track, and can even be used to replace a hardware mixer if all you want to do is play around with the sounds and experiment with the synthesizers.



12 Recording your masterpiece

If you want to record what you're playing, you need to first record-enable each track you want to save, and then press record in Ardour's main transport bar. Record-enabling a track with MIDI data will only record the MIDI data – not the audio. This is highly useful, as it lets you edit out mistakes in timing and pitch. You can even create tracks in this way, using the pencil tool on a vertically expanded MIDI track to program the notes you want the synth to play. Audio tracks will record the audio input and can only be edited in the same way you edit audio files in Audacity.

Being a multi-track recorder, Ardour can record many tracks at the same time, and it will keep the audio separate until you mix them down into a single file. This can be done by selecting Export > Export to Audio File from the Session menu. LV



GPIO ZERO: HARDWARE HACKING SIMPLIFIED

Programming logic meets cardboard and sellotape in our latest Python/Pi project.

LES POUNDER

WHY DO THIS?
- Learn Python 3
- Learn GPIO Zero
- Control hardware using the GPIO pins
- Use programming logic

TOOLS REQUIRED
- A Raspberry Pi running the latest Raspbian release
- An internet connection
- 6 x LED
- 6 x 220Ω resistors
- Male–female jumper cables
- Male–male jumper cables
- A momentary switch
- A buzzer
- Breadboard
- PIR sensor

The Raspberry Pi offers the **RPi.GPIO** library as part of its standard Python install and using this library we can bring physical computing to life with sensors and motors etc. However, for those new to code the **RPI.GPIO** library can be a little tricky to set up and use. Step forward the Raspbery Pi Foundation and one of its key members, Ben Nuttall. Ben has worked with David Jones, of Picamera fame, to create a library that performs similar to **RPi.GPIO** but with less code and easier to understand. So let's learn more about this library via a few projects that will demonstrate how to use the library with LEDs (Light Emitting Diodes), push buttons and sensors, all leading to our final project where we build our own burglar alarm system using a combination of very easy to source and cheap components.

RPi.GPIO vs GPIO Zero

RPi.GPIO was created by Ben Croston, and powers the vast majority of Raspberry Pi hardware projects across the globe. Originally written to help Ben use the Raspberry Pi to control his microbrewery, **RPi.GPIO** is now used in Raspberry Pi projects ranging in complexity from simple LEDs all the way to high altitude/space projects.

RPi.GPIO requires the user to import the library, and typically we rename it in our code to **GPIO** for ease of use. We also import the **time** library to control the pace of our project.

```
import RPi.GPIO as GPIO
import time
```

Next we have to tell the project what GPIO pin layout we're using.

```
GPIO.setmode(GPIO.BCM)
```

For our first project with GPIO Zero we'll do the 'Hello World' of hardware hacking – lighting an LED

Next we create a variable to store the GPIO pin used for our LED, before configuring that pin to be an output that will send power to the LED.

```
led = 17
```

Next we create an infinite loop, which will turn on the LED for half a second before turning it off.

```
while True:
 GPIO.output(led_pin, True)
 time.sleep(0.5)
 GPIO.output(led_pin, False)
 time.sleep(0.5)
```

So the total number of lines of code required to flash an LED on and off with **RPi.GPIO** is nine. Let's see how GPIO Zero compares.

We start by importing the **LED** class from the GPIO Zero library, and we import **pause** from the **signal** library – we'll use that to keep the code active.

```
from gpiozero import LED
from signal import pause
```

Next we instruct our code that we have an LED on GPIO 17.

```
red = LED(17)
```

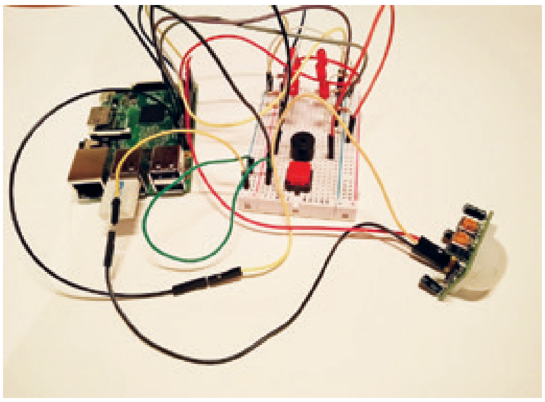
Lastly we use the **blink** function to handle turning the LED on and off, and we end the code by using the **pause()** function to keep our code active; without it the project would blink once before ending.

```
red.blink()
pause()
```

With GPIO Zero it took only five lines of code, so we saved four lines of code and skipped all of the GPIO setup, which can be quite daunting for those new to the Raspberry Pi.

Setting up the software

We used the latest version of Raspbian for this


Here we see the completed project constructed on a breadboard, ready to alert us to a burglar or a dog.

tutorial, which offers GPIO access for all users, and no longer requires launching *Idle* with **sudo**. It also features speed refinements to the overall operating system that will make developing your project more enjoyable. The latest version of Raspbian can be downloaded from **www.raspberrypi.org/downloads**. If you're using an older version of Raspbian you'll need to launch *Idle* via the terminal, by typing.

```
sudo idle3 &
```

Boot your Raspberry Pi to the desktop and open a terminal; you can find a shortcut to the terminal in the menu at the top-left of the screen. With the terminal open we shall now install the dependencies for GPIO Zero. Type the following into the terminal and press Enter to install.

```
sudo apt-get install python-pip python3-pip python-w1thermsensor python3-w1thermsensor python-spidev python3-spidev
```

With those installed, remain in the terminal and type the following to install GPIO Zero using the **pip** package manager for Python 2 & 3.

```
sudo pip install gpiozero
sudo pip-3.2 install gpiozero
```

We'll be using the library for Python 3, but Python 2 is available for those who wish to integrate it into existing projects.

With that installed you can now close the terminal and navigate to the Programming menu, found in the main menu at the top-left of the screen. Click on the Python 3 entry to load *Idle*. With Python 3 *Idle* open click on File > New to open a new blank document. Immediately save your work as **project1.py**.

PROJECT 1

For our first project with GPIO Zero we'll do the "Hello World" of hardware hacking: lighting an LED. Before we write the code let's set up our circuit. We'll need to use our breadboard, an LED, a 220Ω resistor and two female–male leads. One lead will attach to a GND on your Pi, the other to GPIO 17. GPIO Zero uses the Broadcom pin mapping, which is the official standard supported by the Foundation. Please refer to our diagram over the page for how to lay out the components. Components assembled, let's delve into the code.

Our first step is to import two libraries. Firstly we import the LED class from the GPIO Zero library. Secondly we import **sleep** from the **time** library.

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

We next create a variable called **red**, which will contain the location of our LED, which is GPIO17.

```
red = LED(17)
```

Finally we create an infinite loop, which will turn our LED on for one second and then off for a second.

```
while True:
 red.on()
 sleep(1)
 red.off()
 sleep(1)
```

Save your code and click on Run > Run Module. You should now see the LED flash on and off. Congratulations: you have completed your first project with GPIO Zero!

PROJECT 2

Let's try another quick project, this time with a momentary switch (push button) attached to our Raspberry Pi. Remove all of the components from Project 1 and build Project 2 as per the diagram over the page. Create a new file and save it as **project2.py**.

Our goal for this project is to detect a button press and react with a piece of humorous text. So let's start this project by importing the **Button** class from the GPIO Zero library.

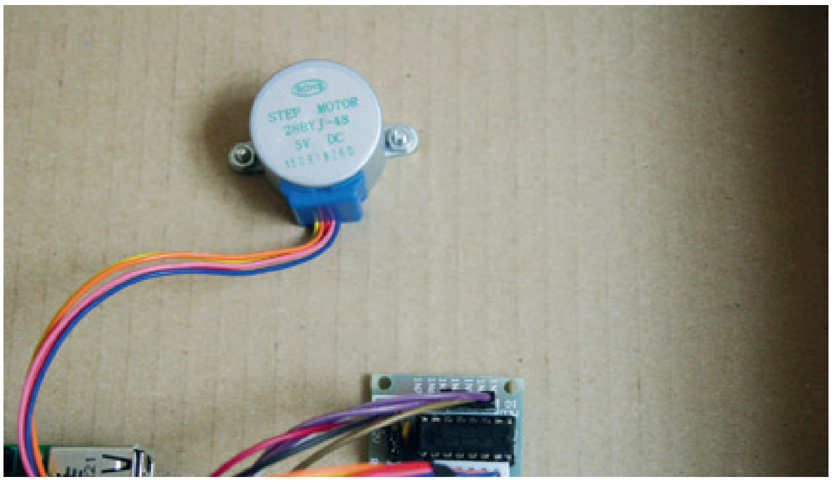
```
from gpiozero import Button
```

We now tell the code where our button is located, which is GPIO 22.

```
button = Button(22)
```

We next create an event that's searching for a button press.

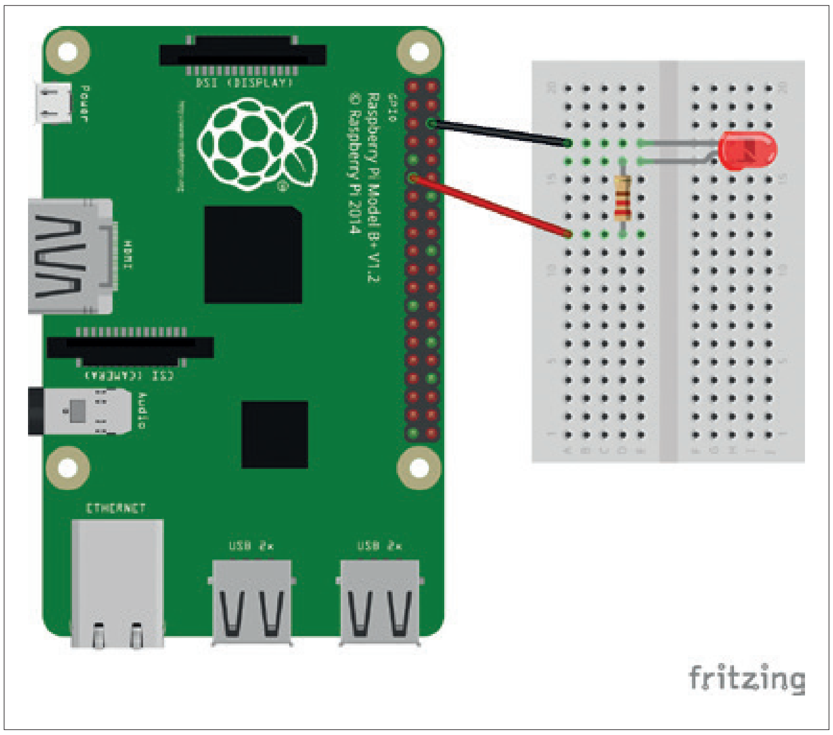
```
button.wait_for_press()
```


A Passive Infrared Sensor (PIR) is an affordable sensor to start your projects. It works really well with the Raspberry Pi and is easy to use.

GPIO Zero add-on boards

In this tutorial we worked with common electronic components that offer a really cost effective entry to hardware hacking. But GPIO Zero is not just about individual components: it can also work with an expanding series of add-on boards from third-party vendors.

GPIO Zero also comes with a series of great classes that can handle controlling common components (we used the **MotionSensor** class to work with a PIR sensor). For those of you who are budding robotics developers there's a **Robot** class that enables robots from common components. Typically a robot control board comes with its own software libraries but by using a typical motor controller, such as an L298N, you can set which GPIO pins are used to control the direction of each motor thus giving you total control of the direction that our robot takes, including some precise spins.



Project 1 is a really simple circuit, so great to learn the ropes with.

Finally we create the text that will be printed to the shell when the button is pressed.

```
print("Hello Linux Voice readers...you are hacking with GPIO Zero")
```

Save your code and click on Run > Run module. Press the button and you'll see text appear on the screen. Congratulations – are you ready for the challenge that is Project 3?

PROJECT 3

In this project we'll use a Passive Infrared Sensor (PIR) to detect movement. A PIR sensor detects movement and when triggered sends a pulse of current via its output pin to a GPIO pin on our Raspberry Pi. In this project we shall trigger an LED to turn on when the PIR sensor detects movement.

Create a new file and save it as **project3.py**. We start as ever by importing the libraries for this project. We will import the **MotionSensor** and **LED** classes from the GPIO Zero library.

```
from gpiozero import MotionSensor, LED
```

Now we tell our code that the PIR sensor is on GPIO 26 and that our LED is on GPIO 17.

```
pir = MotionSensor(26)
led = LED(17)
```

Next we'll create two states for our sensor; the first is for when motion is detected, and will trigger the LED to illuminate for a few seconds (this is controlled by how long the PIR sensor sends current to its GPIO pin). The second state is for when no motion is detected and turns off the LED accordingly.

```
pir.when_motion = led.on
pir.when_no_motion = led.off
```

Save your work and click on Run > Run Module to start your code. Celebrate, as it will trigger the sensor and signify your success with this project.

PROJECT 4

In this project we use GPIO Zero and a few components to build a burglar alarm system. We'll reuse some of the principles used in our previous projects and refine them further. Start by assembling your breadboard so that it looks like the diagram, right. This circuit is a little tricky to build but take your time, check often and you will do great.

Create a new file and immediately save it as **Project4.py** before proceeding. As always we'll start by importing the libraries. Firstly we import the **MotionSensor**, **LED**, **Buzzer** and **Button** classes from the GPIO Zero library. We then import the **sleep** function from the **time** library.

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

Next we create a series of variables that will store the GPIO pin numbers for our passive infrared sensor (PIR), buzzer, button and six LEDs.

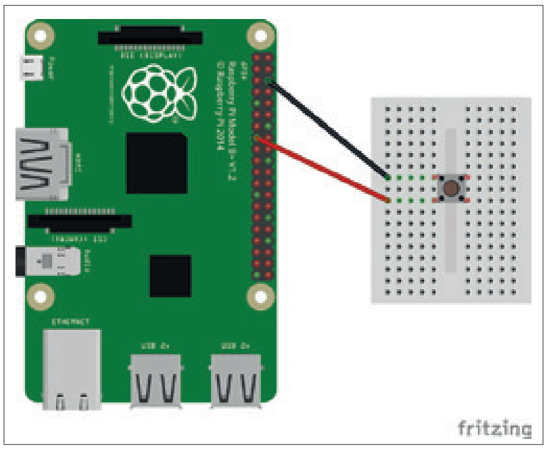
```
pir = MotionSensor(26)
buzzer = Buzzer(19)
button = Button(13)
led1 = LED(17)
led2 = LED(27)
led3 = LED(22)
led4 = LED(10)
led5 = LED(9)
led6 = LED(11)
```

Next we create a list, known as an array in other languages, and in there we store our LED variables. We use the list in our code later.

```
all = [led1,led2,led3,led4,led5,led6]
```

So now we create the first of two functions. Functions group batches of code together, and we can use them by calling the name of the function. Our first function is called **all_on()**, and its purpose is to turn on the buzzer, then use a **for** loop to iterate through all of the entries in our list. Each entry in the list is a reference to a GPIO pin for an LED. So for each LED we instruct it to turn on and then wait for 0.1 seconds before moving on to the next LED.

```
def all_on():
```



Project 2 uses a momentary switch (also called a push button) to act as a method of input. We can trigger any action to occur from this single button.

```
    buzzer.on()
    for i in all:
        i.on()
        sleep(0.1)
```

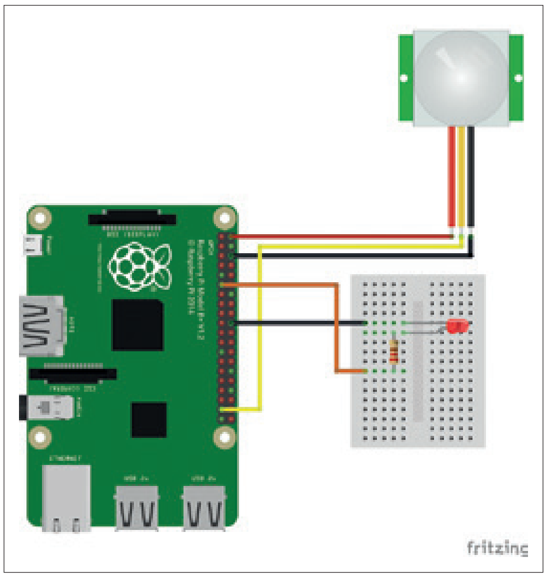
Next we create a second function called **all_off()**, which will ensure that the buzzer and all of the LEDs are turned off.

```
def all_off():
    buzzer.off()
    for i in all:
        i.off()
```

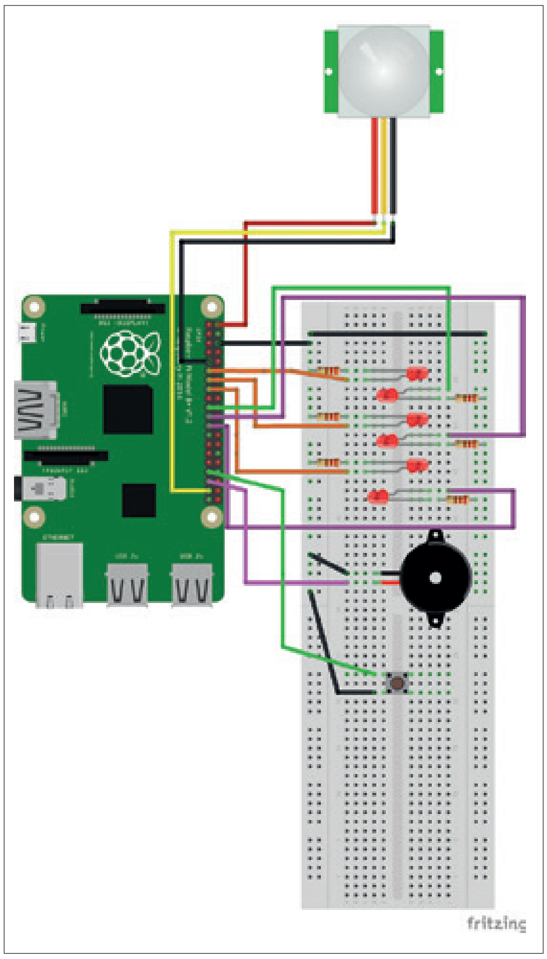
Our final section of code is an infinite loop that will constantly loop round. Inside the loop we add a button-press event that will prevent the PIR sensor from activating until the button is pressed. Once the button is pressed a message is printed to the Python shell, warning you to run away before the alarm is set. The code sleeps for five seconds, plenty of time to hide. Now we create a trigger that instructs the code to react when movement is detected by the PIR sensor. The sensor works by sending current to the Raspberry Pi when movement is detected; GPIO Zero has a special method that will wait for the current on the chosen GPIO pin to change and this acts as a trigger in our code. At rest the PIR sensor sends no current to the GPIO and we instruct the code to ensure that all of the LED and buzzer are turned off using the function we wrote previously.

```
while True:
    button.wait_for_press()
    print("SYSTEM ARMED YOU HAVE 5 SECONDS TO RUN
AWAY")
    sleep(5)
    pir.when_motion = all_on
    pir.when_no_motion = all_off
```

Save your code and click on Run > Run Module to test the code. Press the button and you'll see the warning message appear. It's best to ensure that the sensor is pointing away from you as the it is rather sensitive and has around 180 degrees for detection. Any movement will trigger the sensor, which will call the **all_on()** function and start the alarm process by sounding the buzzer and sweeping through the six LEDs on our breadboard.



In Project 3 we use the PIR sensor to trigger an LED to light up – very handy for silent alerts when tracking burglars.



Project 4 brings together all the components that we used in the previous projects to make one mega project.

It's not the end

GPIO Zero is a welcome project that is sure to do really well. It's still really early days, but this project is already gaining ground and will be included by default in a future Raspbian release.

GPIO Zero is an ideal way to introduce Python programming to children and would be a great bridge between the block-based languages, such as Scratch, and the full implementation of Python.

We can't wait to see what future projects will be powered by this great library. Robotics, weather stations and camera projects are achievable using GPIO Zero, so there are plenty of avenues for future inspiration.

All of the code for this project can be found via our GitHub repository at **http://bit.ly/LV23Code**, or you can download a Zip file containing all of the project files from **http://bit.ly/LV23ZIP**.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

BUILD SIMPLE SCRIPTS WITH **IF THIS THEN THAT**

Create simple programs to manipulate online data and keep you dry in the rain.

BEN EVERARD

WHY DO THIS?

- Stay dry and warm while all around are soaked to the skin
- Make your life easier by harnessing data and getting it to do your bidding

The internet is full of information. News reports, weather forecasts, stock prices, blog posts, cartoons and more are all out there ready for you to read at your leisure. All this information isn't just for you – computers can also use these online news sources.

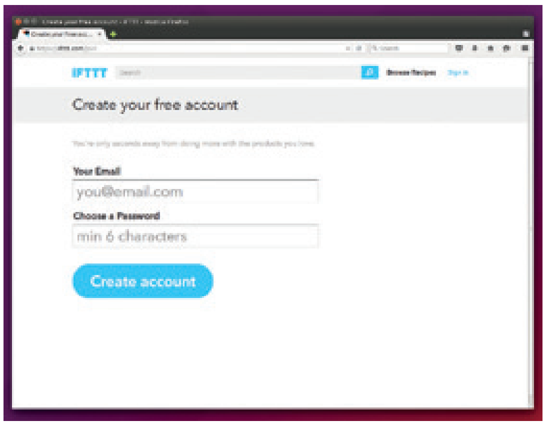
If This Then That (*IFTTT*) is a really simple web app for creating and hosting programs that take an input from the internet and trigger some output. The user specifies the This and That from a predefined list of choices. In this tutorial, we're going to build a simple weather alert that follows the formula: If there's rain forecast tomorrow then send me an alert to remember to take an umbrella when I go out. The whole thing is accomplished with just a few mouse clicks and not even a single line of code.

STEP BY STEP: INSTALL IF THIS THEN THAT

1 Sign up

IFTTT is a hosted service that both provides the tools to create your programs and runs them for you. The first step is to sign up for an account at **http://ifttt.com**. When you first log in, you'll be greeted with a whole range of example programs (known as recipes in *IFTTT* terminology). Browsing these gives you a great sense of what's possible with *IFTTT* – there are recipes to help you stay fit, to monitor your sleep and to find bargains online, among others. As programming languages go, it's very limited compared to almost any other, but what it can do, it does very easily and very well.

If you click on the Channels at the top of the screen, you'll get a list of all the data sources, and clicking on one of these will give details of the most popular published recipes using this channel. There are also categories that show the most popular recipes of a particular type, such as the productivity recipes at **https://ifttt.com/categories/be-more-productive**.

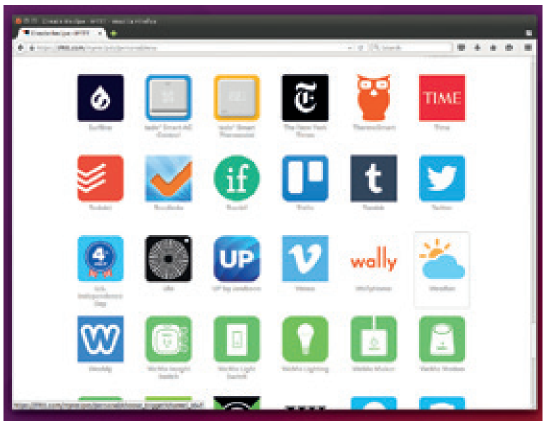


2 If This

To create a new recipe, go to the My Recipes page and click on Create New. *IFTTT* recipes trigger when a particular event occurs, so the first step is always to decide which trigger to use. These triggers come from channels, and there are over 230 of these to choose from. Some of these require specific hardware (including one that works with OpenHAB, which we've looked at in this month's cover feature on building an intert-connected home); others just get information directly from the internet or a smartphone.

Some of the channels we've found most useful are Feeds, which brings any RSS feed into *IFTTT*; Reddit, which creates alerts for topics that you're interested in; and Android Location, which triggers recipes when you enter or leave a particular place.

For our umbrella alert, we're going to use the Weather channel. Click on this icon, and *IFTTT* will take you to the next step.

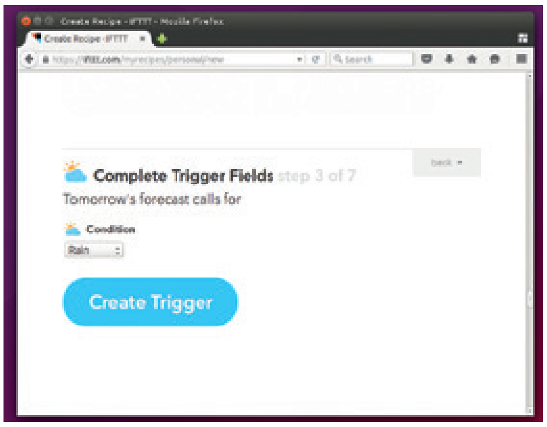


3 Configure Item

Channels aren't single pieces of data, but collections that can be selected from and configured to your needs. The Weather channel will pick up some information automatically (such as your location – to check this is correct, head to **https://ifttt.com/weather**). Other bits need to be configured manually.

In the Weather channel, you can select the type of data you want, such as today's forecast, tomorrow's forecast and specific actions within this forecast. In order for your recipe to behave as you expect, it's important to understand exactly what triggers it.

For our alert, we'll use the channel Tomorrow's Forecast Calls For, which is triggered every time the weather forecast for tomorrow includes a particular weather, and we'll set the condition to be Rain.



Complete Trigger Fields step 3 of 7

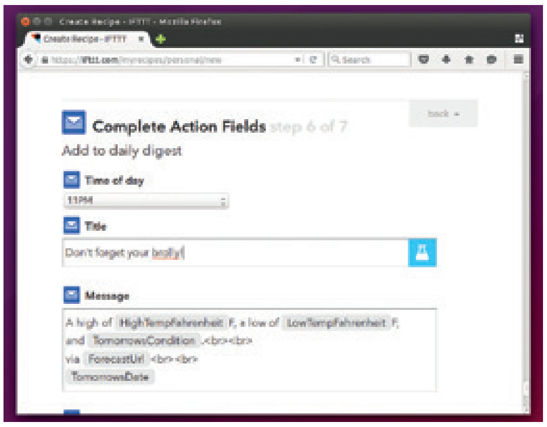
Tomorrow's forecast calls for

Condition: Rain

4 Then That

The final stage of configuration is to define what happens when the event is triggered. Again, there are a whole range of channels that you can use to define the behaviour you want.

Our weather event happens every time the forecast changes to include rain. However, we don't want a constant flood of notifications every time the forecast is updated. Instead, we just want a single message that we can get first thing in the morning to tell us what's going on. For this, the best option is an Email Digest. This channel stores up notifications and sends them out at a particular time that you can specify. Since all our notifications are about tomorrow, we'll set this to run late one night (11pm) and it'll email us if there's been rain forecast for the day.



Complete Action Fields step 6 of 7

Add to daily digest

Time of day: 11PM

Title: Don't forget your brolly!

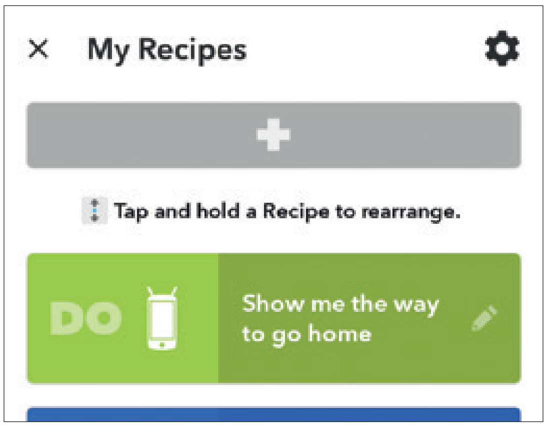
Message: A high of HighTempFahrenheit F, a low of LowTempFahrenheit F, and TomorrowsCondition .

 via ForecastUrl

 TomorrowsDate

5 Do Button, Note and Camera

Recipes run automatically according to a specific set of rules. These work well in many circumstances; however, sometimes we need to be able to trigger a rule whenever we want. For this, there's the DO button. This is a smartphone app that enables you to create recipes that trigger whenever you press a button. The DO button app is available from the Google Play and iTunes stores. Once it's installed, you can place a widget on your home screen to trigger a IFTTT action. The button can also pass location data to the That section of your recipe. If you need more information, you can also install the Note and Camera apps. These, as you've probably guessed, enable you to add small amounts of text or images to your recipes.

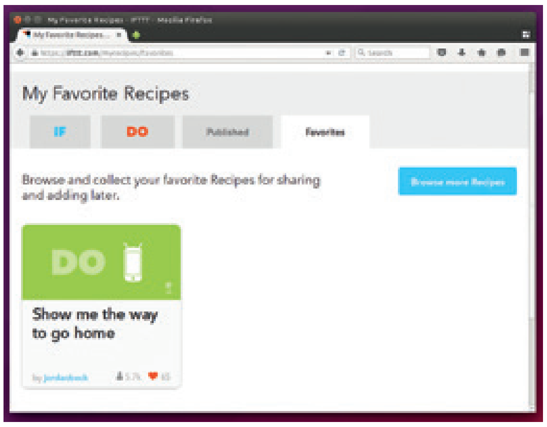


My Recipes
Tap and hold a Recipe to rearrange.
DO
Show me the way to go home

6 Managing recipes

At the start of this tutorial, you had a look at some of the most popular recipes that other users had shared. When you create a useful *IFTTT* recipe, it's a good idea to publish it to enable other users to take advantage. This is done in My Recipes > Edit > Publish.

IFTTT gives you a few tools to help you manage recipes in the My Recipes page. If there's one that you want to keep, but don't want to have running at the moment, you can turn it off. This leaves it in the My Recipes page, but deactivates it. If you want to get rid of a recipe, you can delete it permanently by going to Edit > Delete. You can also favourite recipes (by clicking on the heart icon) that you like, but don't want to install at the moment. LV



My Favorite Recipes

Browse and collect your favorite Recipes for sharing and adding later.

KEEP TABS ON THE WEB WITH RSS FEEDS

Getting information out of the web is like drinking from a firehose – so use a filter!

MARCO FIORETTI

WHY DO THIS?

- Get the headlines without having to check the BBC every 10 minutes
- Group news feeds into lists based on your areas of interest
- Filter, process and redistribute news of all sorts as efficiently possible.

The greatest damage that Facebook *et al* may have done is that they've made the world forget that RSS exists. RSS (the acronym stands for Really Simple Syndication), as well as its successor and competitor Atom, is a file format designed to share and distribute content, usually over the internet. In this tutorial we explain what RSS and Atom feeds are, why sharing and distributing content is so important, and how to use them to read, create, process, distribute and reuse content.

We wanted this to be a tutorial for absolute beginners. All you need to start playing with the tricks in these pages are a really basic understanding of how scripting works and (possibly) Linux computer with a text editor, a command line and an internet connection. In order to keep the tutorial simple, all the scripts shown here are simplified ones. They work as described on our computers, with all the RSS/Atom feeds we tried – but they are not optimised for performance or robustness, and therefore may not work on computers or feeds with non-ASCII locales or contents. Second, even if we almost always only say "RSS" for brevity, most of what you'll read here also applies to Atom.

RSS and Atom are XML (eXtensible Markup Language) formats – that is plain text, but filled with ugly-looking markup tags. Both formats were developed to let people know all the most recent articles that they could find on a certain website, without loading its home page in a browser. The "RSS feed" file of a website, which is automatically updated every time new content is added, contains titles, dates and links to the full text of all new articles. Yes, we know: said that way, an RSS feed seems just a stripped-down version of a home page, without any real reason to exist. But RSS really is a big deal, for at least three reasons:

> With the right software it's easy to aggregate, filter and read as many independent feeds as you want

1 Efficiency. Sure, RSS was built to distribute headlines of articles, usually with excerpts, but nothing forbids other uses. As we will see shortly, you can use RSS to distribute pretty much everything that can be expressed by text. Everything! For the same reason, it is easy to "reuse" data you got via RSS in all sort of places.

At the same time, with the right software it's easy to aggregate, filter and read as many independent feeds as you want all together, in one compact window of your screen, without all the overhead, distractions and other annoyances you'd be forced to endure on social networks that offer similar services. Trust us: RSS can be a huge time saver.

2 The second big reason for RSS is to control for yourself what you see and how you see it. Do you want to stay inside Facebook's filter bubble all the time? RSS comes to you from the actual sources, without intermediaries.

3 The third reason for RSS is to avoid being controlled – to read what you want, without anybody tracking it. RSS alone cannot prevent tracking, but is a necessary part of privacy-enhancing strategies.

This is the magic RSS symbol – websites without it aren't serving their content as well as they should.

RSS structure

RSS 2.0, the current version of the format, is relatively limited (see the What's New In Atom box for a comparison, but also pretty simple to use. It is also the most popular one, if nothing else because it still is the default format in many content management systems. Listing 1 shows some parts, edited for clarity, of the Linux Voice atom feed, which is always waiting for you at the address **http://www.linuxvoice.com/feed/**:

```
Listing 1:
<channel>
    <title>Linux Voice</title>
    <atom:link href="http://www.linuxvoice.com/feed/"
rel="self" type="application/rss+xml" />
    <link>http://www.linuxvoice.com</link>
    <description>The magazine that gives back to the
free software community</description>
    <lastBuildDate>Mon, 19 Oct 2015 12:50:18 +0000</
lastBuildDate>
...
<item>
<title>Create your own desktop environment</title>
<link>http://www.linuxvoice.com/create-your-own-
desktop-environment/</link>
<pubDate>Tue, 13 Oct 2015 13:45:27 +0000</pubDate>
<dc:creator><![CDATA[Mike Saunders]]></dc:creator>
<category><![CDATA[Featured story]]></category>
<description>What to do if KDE, Gnome and Xfce don't
float your boat? Build your own custom desktop
environment, of course!...</description>
...
</item>
```

The first part contains a name, source link, description and other metadata about the whole feed. After that, there is a list of items, each containing all the information related to one article: title, link to full text, author, publication date, excerpt and so on. All in nice, plain text, easy to read and process in many different ways.

Read RSS anywhere

The easiest way to read RSS feeds is to catalog and filter all of them together inside a dedicated RSS aggregator. The image below shows what the raw RSS "source code" in the code above looks like from within *Liferea*, the fantastic Free Software news reader.

The left-hand side of the same figure confirms what we already said: with RSS, one simple, compact interface lets you collect, sort and filter as you wish a lot of totally unrelated news sources.

RSS aggregators such as *Liferea*, which are sometimes also called feed readers, exist for all platforms and tastes. On Linux and Unix-related desktops, including Mac OS X, you may use the command-line program *Newsbeuter* (**http://newsbeuter.org**). This fine piece of mimimalist software calls itself "the *Mutt* of RSS feed readers", and for good reasons: you can completely control it with the keyboard, and it has many functions to filter, sort, tag and search the content of each feed. There are also several RSS aggregation apps or services for mobile platforms, including, *Reader+* or *gReader*. Web-based Free Software like *Tiny Tiny RSS* (aka *TT-RSS*, **https://tt-rss.org**) lets you aggregate and read RSS feeds directly in your browser, or in its official Android app. *TT-RSS* runs whenever there is a web server with PHP support, and access to a *PostgreSQL* or *MySQL* database. This may be your own Linux laptop, or any free or entry-level hosting account in a data centre.

Installation of *TT-RSS* is as simple as it can be for a web-based application. Unpack the source files in a folder accessible to the web server, edit the configuration file to set the database parameters, point your browser to the root folder of the installation and follow the instructions. A single installation of *TT-RSS* may serve as many users as you wish, each with his or her own password, graphic theme, tagging system and collection of feeds.

TT-RSS may also be a great assistant for RSS-based content curation. Its users, in fact, can aggregate and filter the items in all the feeds that they download with *TT-RSS*. The resulting lists of articles, which of course vary every day as the content of their sources varies, can be published from within *TT-RSS* as new, dynamic RSS feeds that everybody can download and read directly, in any other RSS aggregator.

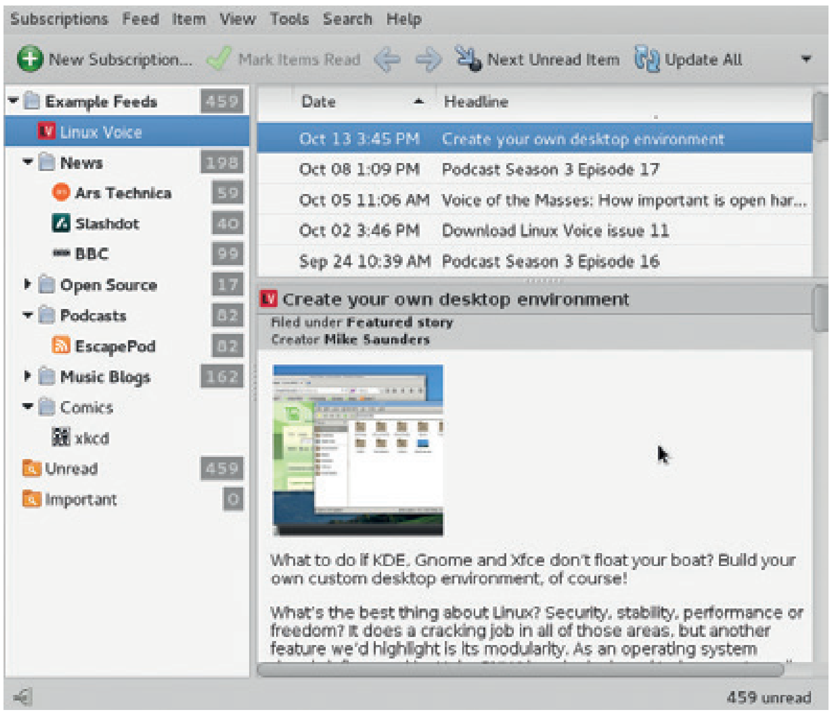
PRO TIP

In these pages we've only used Python, but RSS libraries like the ones shown below also exist for Perl and other scripting languages which, depending on your needs, may be better than Python.

Create your own RSS feeds

So far, we have summarised the main, standard ways to use RSS feeds produced elsewhere. Now the fun begins. Let's see how, with a bit of scripting, you may consolidate practically every kind of information as an RSS feed and reuse it as and where you wish. Let's start with the **rssbuilder** Python script of Listing 2:

Figure 1: RSS aggregators and readers such as *Liferea* have made it possible to fetch and browse hundreds of headlines at a time.

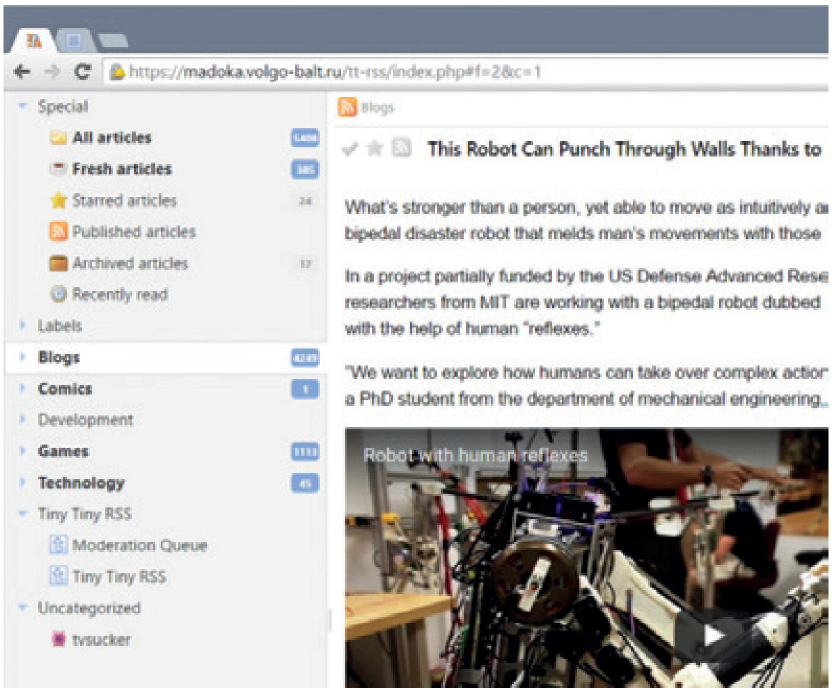


Date	Headline
Oct 13 3:45 PM	Create your own desktop environment
Oct 08 1:09 PM	Podcast Season 3 Episode 17
Oct 05 11:06 AM	Voice of the Masses: How important is open har...
Oct 02 3:46 PM	Download Linux Voice issue 11
Sep 24 10:39 AM	Podcast Season 3 Episode 16

Create your own desktop environment
Filed under Featured story
Creator Mike Saunders

What to do if KDE, Gnome and Xfce don't float your boat? Build your own custom desktop environment, of course!

What's the best thing about Linux? Security, stability, performance or freedom? It does a cracking job in all of those areas, but another feature we'd highlight is its modularity. As an operating system



This Robot Can Punch Through Walls Thanks to

What's stronger than a person, yet able to move as intuitively a
bipedal disaster robot that melds man's movements with those

In a project partially funded by the US Defense Advanced Rese
researchers from MIT are working with a bipedal robot dubbed
with the help of human "reflexes."

"We want to explore how humans can take over complex actior
a PhD student from the department of mechanical engineering,

Figure 2: Is this a website? Is this an RSS reader? Actually it's both. *Tiny Tiny RSS* is a web-based application that lets multiple users read, filter, create and share RSS feeds as they wish.

Listing 2:

```
 1 #! /usr/bin/python
 2
 3 import sys
 4 reload(sys)
 5 import datetime
 6 import PyRSS2Gen
 7
 8 sys.setdefaultencoding('utf8')
 9
10 rss = PyRSS2Gen.RSS2(
11    title = "Bare DEMO Feed for Linux Voice",
12    link = "http://www.linuxvoice.com/samplefeed",
13    description = "just a bare feed to demonstrate the power of RSS scripting",
14
15    lastBuildDate = datetime.datetime.now(),
16    )
17
18 fileHandle = open(sys.argv[1], 'r')
19
20 for line in fileHandle:
21    fields = line.split('|')
22    item = PyRSS2Gen.RSSItem(
23      title = fields[2],
24      link = fields [3],
25      description = 'just a demo!',
26      pubDate = fields[0]
27      )
28    rss.items.append(item)
29
30 fileHandle.close()
31
32 rss.write_xml(open(sys.argv[2], "w"))
```

There are tons of applications that can catch and use RSS feeds, including WordPress and all decent CMSes

The purpose of the script is to transform lists of news in plain text format (with one "news" per line composed of several fields separated by the pipe (|) character) into one, standard RSS feed that any aggregator might read.

The first eight lines of the script load several Python modules and (re)configure system variables, like character encoding, that the script needs.

Lines 10–16 create a new RSS object, and define its title, source link, description and creation date. The loop in lines 20–27 reads the text file passed as the first argument in line 18, splits each line using the pipe character as separator (line 21), puts each resulting field in the corresponding one of a new RSS item, and finally appends it to the RSS object (line 28).

The last two commands close the input file, and write a whole RSS feed to another file, whose name is the second argument given to the script. For example, if you ran **rssbuilder** in this way inside a terminal:

```
#> rssbuilder allmynews.txt output.xml
```

using an "allmynews.txt" file with this content:

```
2015-10-24 07:16:25+00:00 |bbc|BBC VIDEO: Is this the iceberg that sank the Titanic?|http://www.bbc.co.uk/news/uk-34625510
2015-10-15 09:06:21+00:00 |mycomputer|Your print queue is stuck|http://localhost:631
2015-10-13 13:45:27+00:00 |linuxvoice|From LV: Create your own desktop environment|http://www.linuxvoice.com/create-your-own-desktop-environment/
2015-10-02 13:46:39+00:00 |gmail|You got email from Aunt Judy|http://www.gmail.com
2015-10-01 08:39:29+00:00 |mycomputer|Your hard drive is almost full!|http://localhost
```

Inside any RSS reader the output would look more or less like what's inside *Akregator* in figure 3. That screenshot highlights the real power of RSS. Email notifications, web pages, computer statuses... RSS makes it easy to consolidate, distribute and use information of all sorts, from every conceivable source. Isn't that cool?

Sure, this only happens after you've written down all the initial information as in Listing 3. But that's not as hard as you might fear. If the original content comes from other RSS feeds, you can parse and filter it as you want, with the techniques that we'll show you in a moment. Non-RSS sources, like the email and disk notifications of Figure 3, are just as easy to format in the same way, even if that would be a topic for a separate tutorial.

It's now time to look at how to read and process already existing RSS feeds. One way is the 22-line script called **rss-fetcher** in Listing 4:

Listing 4

```
 1 #! /usr/bin/python
 2
 3 import sys
 4 import feedparser
 5 import socket
 6 from dateutil import parser, tz
 7
 8 timeout = 120
 9 socket.setdefaulttimeout(timeout)
10
11 feed_name = sys.argv[1]
12 feed_url  = sys.argv[2]
13 myrss = feedparser.parse(feed_url)
14 for s in myrss.entries:
15       title = unicode(s.title).encode("utf-8")
16       title = title.replace('\n', ' ')
17       title = title.replace('\r', ' ')
18       title = title.replace('|', '--')
19       datepublish  = s.published
20       dt = parser.parse(datepublish)
21       print(dt),
22       print "|" + feed_name + "|" +  title  + "|" +
unicode(s.link).encode("utf-8") + "\n"
```

As in the other script, the first nine lines load some libraries, which are all installable with the usual software management tools of your Linux distribution. The two arguments that **rss-fetcher** needs (and loads in lines 11 and 12), are an arbitrary feed name and the URL of an RSS feed. Line 13 does the real work, downloading the feed and saving it into an object called **myrss**, with the methods provided by the Python **feedparser** library.

The loop starting in line 14 reads, formats and prints out the most important fields of each item.

Download, parse and filter feeds

As a practical example, this is what we got when we ran **rss-fetcher** on the Linux Voice feed (both URLs and titles are trimmed for readability):

```
#> rss-fetcher lv http://www.linuxvoice.com/feed/ >
linuxvoice.txt
#> cat linuxvoice.txt
2015-10-19 12:49:12+00:00 |lv|Voice of the Masses: Who is
your Linux or Free Software hero?|www.linuxvoice.com/
voice-of-the-masses...
2015-10-13 13:45:27+00:00 |lv|Create your own desktop
environment|http://www.linuxvoice.com/create-your-
own-desktop-environment/
2015-10-08 11:09:11+00:00 |lv|Podcast Season 3 Episode
17|http://www.linuxvoice.com/podcast-season-3-
episode-17/
2015-10-05 09:06:21+00:00 |lv|Voice of the Masses: How
important is open hardware?|http://www.linuxvoice.com/
voice-of-the-masses...
2015-10-02 13:46:39+00:00 |lv|Download Linux Voice
issue 11|http://www.linuxvoice.com/download-linux-
voice-issue-11/
```

Now, please take a moment to imagine what you might get by combining two scripts like these. Being able to transform an RSS feed (or any other information) in a series of pipe-delimited lines inside a plain text file, and to transform files like that into standard RSS feeds means that you can, with just a bit more of scripting:

1. Save many feeds, from many different sources, in plain text files that are perfect for further, easy processing;
2. Combine, filter and/or sort the content of all those files in any way you wish;
3. Format the result(s) of step 2 into new standard feeds, reusable by everybody or, alternatively;
4. Embed the results of step 2 in any part of your desktop or websites.

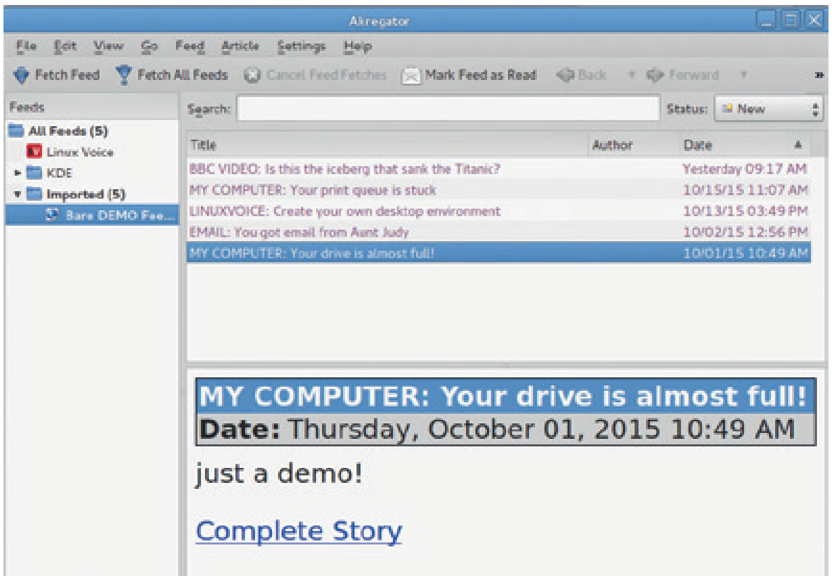
Step 4 is described in the last part of the tutorial, but first let us explain step 2 a bit better. Imagine you've saved many different feeds in as many text files called **feed-01.txt**, **feed-02.txt** and so on, using the **rss-fetcher** script. At that point, commands like these:

```
#> grep '2015-10-15' feed*.txt | sort > today.txt
#> grep -i linux feed*.txt | sort > linux.txt
#> grep -i '|KEYWORD|' feed*.txt | sort > KEYWORD.txt
```

will extract, sort by date and save the news from today (**today.txt**) or contain a specific word in the title (**linux.txt**) or fit into whatever category you passed to **rss-fetcher** (**KEYWORD.txt**). At that point, it would be quite easy to pass those files to **rss-builder**, or to reuse them in the other ways described below all through one script that does the whole work automatically. The hardest part may be to figure out exactly, with pen and paper, all that you actually want to filter or process, and how.

Reuse RSS everywhere. Really everywhere

There are tons of end-user applications that can fetch and display RSS feeds. *WordPress* and all other decent content management systems, for example, all have modules that let webmasters automatically embed



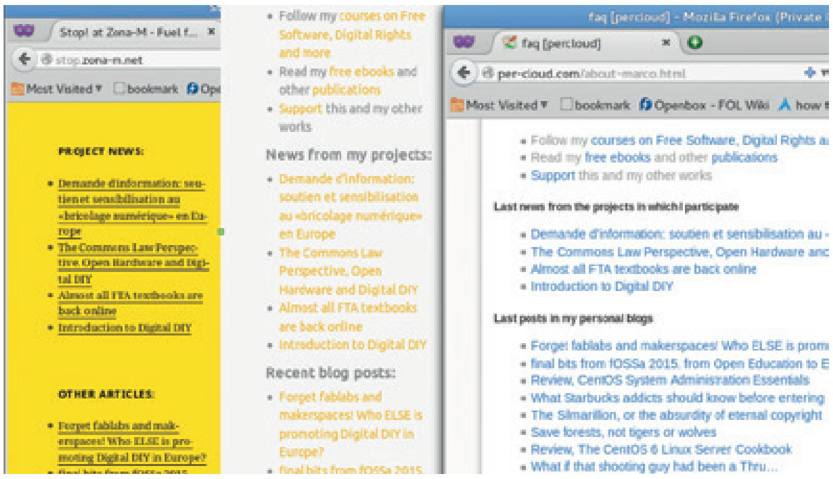
Title	Author	Date
BBC VIDEO: Is this the iceberg that sank the Titanic?		Yesterday 09:17 AM
MY COMPUTER: Your print queue is stuck		10/15/15 11:07 AM
LINUXVOICE: Create your own desktop environment		10/13/15 03:49 PM
EMAIL: You got email from Aunt Judy		10/02/15 12:56 PM
MY COMPUTER: Your drive is almost full!		10/01/15 10:49 AM

MY COMPUTER: Your drive is almost full!
Date: Thursday, October 01, 2015 10:49 AM

just a demo!

Complete Story

Figure 3: International news, computer warnings, email notifications... everybody can integrate whatever they want into RSS feeds, as long as it is formatted in the right way



Stop! at Zona-M - Fuel f...

stop.zona-m.net

Most Visited ▾ bookmark Ope

PROJECT NEWS:

- Demande d'information: soutien et sensibilisation au «bricolage numérique» en Europe
- The Commons Law Perspective, Open Hardware and Digital DIY
- Almost all FTA textbooks are back online
- Introduction to Digital DIY

OTHER ARTICLES:

- Forget fablabs and makerspaces! Who ELSE is promoting Digital DIY in Europe?
- final bits from fOSSa 2015

- Follow my courses on Free Software, Digital Rights and more
- Read my free ebooks and other publications
- Support this and my other works

News from my projects:

- Demande d'information: soutien et sensibilisation au «bricolage numérique» en Europe
- The Commons Law Perspective, Open Hardware and Digital DIY
- Almost all FTA textbooks are back online
- Introduction to Digital DIY

Recent blog posts:

- Forget fablabs and makerspaces! Who ELSE is promoting Digital DIY in Europe?
- final bits from fOSSa 2015

faq [percloud] - Mozilla Firefox (Private

faq [percloud]

per-cloud.com/about-marco.html

Most Visited ▾ bookmark Openbox - FOL Wiki how

- Follow my courses on Free Software, Digital Rights a
- Read my free ebooks and other publications
- Support this and my other works

Last news from the projects in which I participate

- Demande d'information: soutien et sensibilisation au -
- The Commons Law Perspective, Open Hardware anc
- Almost all FTA textbooks are back online
- Introduction to Digital DIY

Last posts in my personal blogs

- Forget fablabs and makerspaces! Who ELSE is prom
- final bits from fOSSa 2015, from Open Education to E
- Review, CentOS System Administration Essentials
- What Starbucks addicts should know before entering
- The Silmarillion, or the absurdity of eternal copyright
- Save forests, not tigers or wolves
- Review, The CentOS 6 Linux Server Cookbook
- What if that shooting guy had been a Thru...

Figure 4: Three websites, with three wildly different architectures, but all embedding the same RSS feed, or what looks like an RSS feeds, automatically synchronized every time new content comes in. How can that be possible?

and display feeds in all their pages. However, being able to play with RSS and Atom as we've just seen opens up a much bigger world of opportunities. Who says that you should see those feeds only in places (be they desktop clients or website modules) that were specifically written for that purpose?

Think of your desktop. Depending on your own habits, and on how your own brain works, it may be very efficient to read RSS feeds in your taskbar, or in menus that appear when you click on your wallpaper. Or have your computer play some alarm when there are news items on some specific topic. The latter case is the easiest to handle. Some shell code like this:

Listing 5:

```
1 #! /bin/bash
2 NEWS=`grep -i -c Linux today.txt`
3 if [ "$NEWS" -gt "0" ]
4 then
5    paplay there-is-news.mp3
6 fi
```

will play the MP3 file specified in line 5 whenever **$NEWS** is greater than zero, that is whenever there is at least one line containing the Linux keyword in the file where you collected today's news. As it stands, the code is not complete, as it will play again for the same news, but you get the idea.

RSS in every page, of every website

What you see in Figure 4 are, from left to right, parts of three real Web pages, each managed in a totally different way from the others. The yellow sidebar on the left is part of a *WordPress* blog. The central, white section is the sidebar of another website, created with the *Mynt* static website generator (**http://mynt.uhnomoli.com**). The rightmost window in Figure 4 shows the body of a static HTML page, in yet another website. What makes them relevant for this tutorial is that those very different websites are all being automatically updated with the same snippets of text, created with more complex versions of the scripts above. Every few hours, one cron job downloads and converts both RSS feeds and non-RSS news, merging them all into one text file formatted as in Listing 2. That file is then converted into raw HTML with a script similar to **rss-builder**, and then "injected" in the places you see in Figure 4, each time with a technique compatible with the target website. In the first case, the code is written into a previously defined widget using the **wp-cli** command line interface for *WordPress* (**http://wp-cli.org**). In the other cases, a placeholder string is replaced in the *Mynt* templates, and in the body of the HTML page.

Conclusion? Go RSS, of course!

After reading this tutorial you may agree with us that much of what you get in a Facebook wall, or Twitter timeline, is not so different from a glorified collage of RSS-like updates from several, otherwise unrelated sources (a superbly glorified and well-done collage, we'll grant you that). In any case, we hope that these few pages have made you like the power and flexibility of RSS and Atom, and given you lots of wild ideas to play with. Please tell us about them!

Marco Fioretti is a Free Software advocate and open data campaigner who has advocated FOSS all over the world.

What's new in Atom?

RSS took over the web in the early 2000s because it was very simple to use, and solved a big problem of the pre-Twitter, pre-Facebook era: getting updates from all over the web, without having to open hundreds of different websites. The success of RSS was big enough, however, to highlight a few flaws that were overlooked by its designers.

The solution to those problems is the Atom Syndication Format (**https://tools.ietf.org/html/rfc4287**). This is an IETF standard with a registered MIME type, and mandatory inclusion of its title and the URL from which it can be retrieved. MIME types are a standard way to classify file types. MIME compliance, together with the mandatory inclusion of the source URL, guarantee autodiscovery – meaning that any standards-compliant browser or aggregator will immediately find by themselves the URL of the feed of any Atom-compliant website, even if all you do is point those programs to its home page, or to a local copy of their feed on your disk.

For developers, Atom software libraries are also much more modular than the corresponding ones for RSS, and are therefore more reusable in other projects. Atom is also better than RSS when it comes to character encoding, content format specification and post modification timestamps. In an Atom feed it's much easier to hande non-ASCII characters and make sure that you do not miss new versions of an article that you already saw. Better content specifications also guarantee that Atom feeds can include more types of content than RSS besides text (audio, images etc...), in a way that will not confuse software clients.

FLOSSUK Spring 2016

15-17th March, Mary Ward House, London

http://bit.ly/flossuk2016 : http://bit.ly/flossuk2016lanyrd

Perl Training 2016

8-12th February, Ambassadors Hotel, London

http://bit.ly/flossperl2016

JOIN FLOSSUK TODAY!

http://bit.ly/flossukmembers

flossUK

OPEN

TECHNOLOGY

office@flossuk.org, www.flossuk.org

Free/Libre Open Source Systems

Open Software - Open Hardware

Open Data - Open Rights

BARE METAL: ARM ASSEMBLY LANGUAGE

Hack code on your Raspberry Pi (or thousands of other devices) at the lowest level.

MIKE SAUNDERS

WHY DO THIS?
- Write blazingly fast code
- Learn another CPU architecture
- Start to build your own OS!

From issues 12 to 16 we ran a series on x86 (PC) assembly language, and we received lots of positive feedback. Although assembly may not seem especially relevant in today's world, where most software is written in high-level languages, it's still mightily useful in the embedded space, where you don't have much RAM or storage space to play around with. Being able to use raw CPU instructions helps if you want to optimise certain routines that are run thousands of times a second – like in video games or physics simulations. Oh, and learning assembly just for pure geektasticness is great fun too.

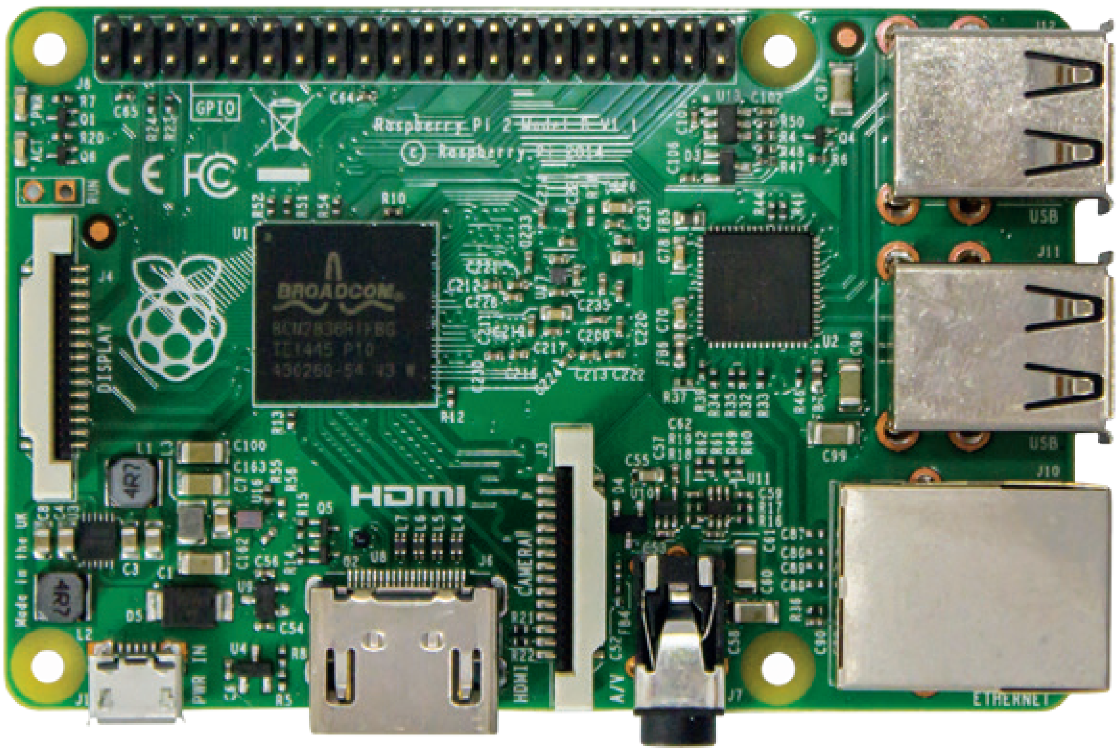
Our x86 assembly tutorials went through the basics of writing code on Linux, and we then moved on to creating a (very simple) operating system and adding graphics effects. If you're new to Linux Voice and want to read the series, you can access all back issues in DRM-free digital format by buying a print or digital subscription at **http://shop.linuxvoice.com**.

Anyway, some readers have asked us to expand on the x86 series of tutorials by covering ARM chips. These are the processors used inside the stonkingly popular Raspberry Pi, along with pretty much every smartphone in the world and thousands of other devices. Over 50 billion ARM chips have been produced since the first designs were crafted by Sophie Wilson and her team at Acorn back in 1983. And while x86 chips have dominated the PC space since the 1980s by offering more raw performance than ARM equivalents, ARM still offers many advantages: less power consumption and a more elegant instruction set, to name two.

Setting up

For this tutorial we'll be focusing on the Raspberry Pi, as it's pretty much the perfect environment for ARM assembly programming: it's cheap (now just $5!!), easy to get hold of, and runs Linux (and therefore the superb GNU toolchain) like a champ.

If you don't have a Pi but still want to explore ARM coding, you could try installing an ARM Linux distribution on the *Qemu* PC emulator. For example, the Fedora project has some instructions for running older ARM versions of the distro in *Qemu* here: **https://fedoraproject.org/wiki/Architectures/ARM/HowToQemu**.



You don't need expensive development boards to hack ARM code – an off-the-shelf Raspberry Pi is ideal.

If you're running Raspbian on a Pi, the two utilities you'll need are **as** (the assembler, which converts assembly language source code into binary code) and **ld** (the linker, which creates the resulting executable file). Both of these are provided in the **binutils** package – so they may already be installed by default. Of course, you'll need a good text editor as well; we always recommend *Vim* for coding, but it has a steep learning curve so *Nano* or a graphical editor will work fine as well.

Ready to go? Type this in and save it as **myfirst.s**:

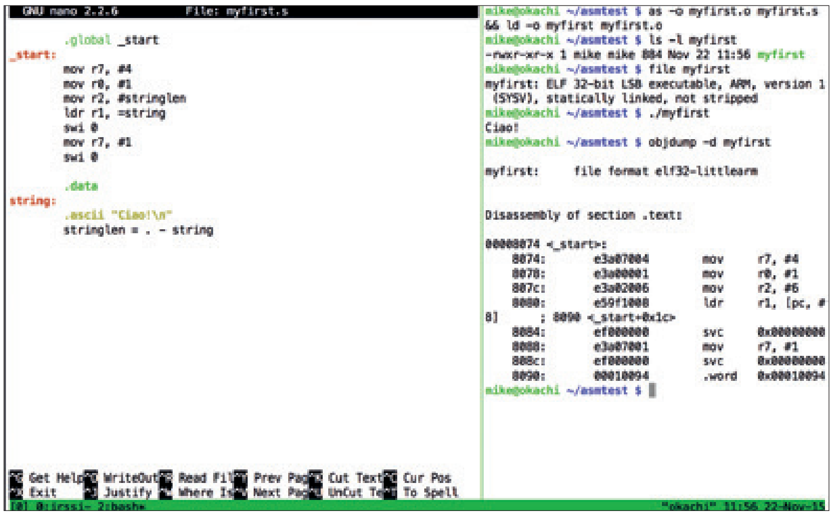
```
        .global _start
_start:
        mov r7, #4
        mov r0, #1
        ldr r1, =string
        mov r2, #stringlen
        swi 0

        mov r7, #1
        swi 0

        .data
string:
        .ascii "Ciao!\n"
        stringlen = . - string
```

This program simply prints the string "Ciao!" to the screen, and if you followed our x86 assembly series, some of it may look familiar to you. But there are many differences between x86 and ARM – and also in the syntax used in the source code – so we'll go through it in detail.

But before that: to assemble the code and link the resulting object file into an executable file, use this command:

```
as -o myfirst.o myfirst.s && ld -o myfirst myfirst.o
```

Now you can run the program in place with **./myfirst**. You'll notice that the program is very small at around 900 bytes – if you did the same thing in C using **puts()**, the binary would be over five times bigger!



```
GNU nano 2.2.6          File: myfirst.s

        .global _start
_start:
        mov r7, #4
        mov r0, #1
        mov r2, #stringlen
        ldr r1, =string
        swi 0
        mov r7, #1
        swi 0

        .data
string:
        .ascii "Ciao!\n"
        stringlen = . - string
```

^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
[0] 0:irssi- 2:bash*
"okachi" 11:56 22-Nov-15

```
mike@okachi ~/asmtest $ as -o myfirst.o myfirst.s
&& ld -o myfirst myfirst.o
mike@okachi ~/asmtest $ ls -l myfirst
-rwxr-xr-x 1 mike mike 884 Nov 22 11:56 myfirst
mike@okachi ~/asmtest $ file myfirst
myfirst: ELF 32-bit LSB executable, ARM, version 1
 (SYSV), statically linked, not stripped
mike@okachi ~/asmtest $ ./myfirst
Ciao!
mike@okachi ~/asmtest $ objdump -d myfirst

myfirst:     file format elf32-littlearm


Disassembly of section .text:

00008074 <_start>:
    8074:       e3a07004        mov     r7, #4
    8078:       e3a00001        mov     r0, #1
    807c:       e3a02006        mov     r2, #6
    8080:       e59f1008        ldr     r1, [pc, #
8]      ; 8090 <_start+0x1c>
    8084:       ef000000        svc     0x00000000
    8088:       e3a07001        mov     r7, #1
    808c:       ef000000        svc     0x00000000
    8090:       00010094        .word   0x00010094
mike@okachi ~/asmtest $
```

Here we're SSHed into a Raspberry Pi, running *Tmux* to split the screen, editing code in *Nano*, and assembling it. Geek bliss.

How it all works

The first two lines here aren't CPU instructions but directives to the assembler and linker. Every program needs a marked starting point called _start, which just so happens to be at the top of the source code in our

The Pi is the perfect environment for ARM assembly programming – it's cheap, easy to get hold of and runs Linux

case. So we confirm with the linker that yes, execution should begin right at the top of our file, and we're ready to go.

In the next instruction we put the number 4 into the **r7** register. (If you've never done any assembly

Write your own Pi operating system

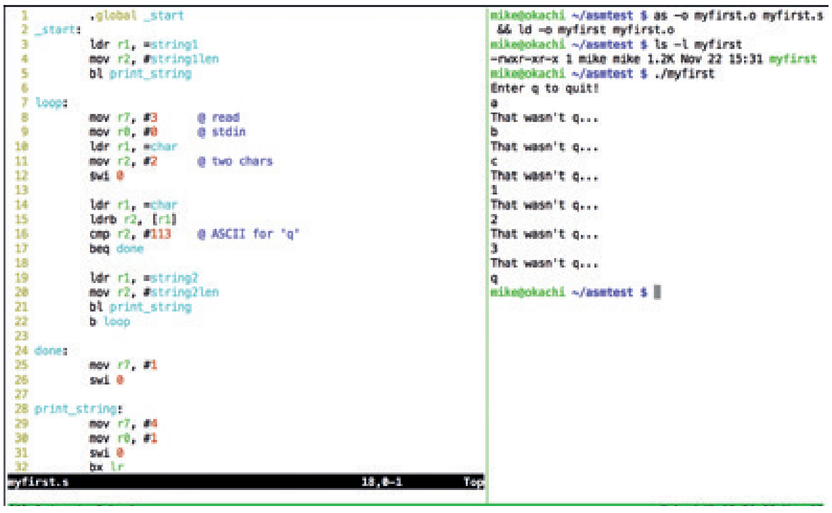
If you followed our x86 assembly language series, you'll recall the moment of sheer awesomeness when you booted up your very first bare metal code, showing a message on the screen without the help of Linux or any other operating system. We then expanded this to include a simple command line and a way to load and run programs from the disk, in effect creating a very simple OS that could be expanded further. This was all very cool, but the process was made much simpler thanks to the BIOS – it provided simplified access to the screen, keyboard and floppy disk drive.

With the Raspberry Pi, you don't have these BIOS routines to assist you, so you have to write drivers from scratch, which is really a pain in the rear when you'd rather be focused on cool stuff like printing things on screen and running programs. Still, there are a number of guides on the web that take you through the initial steps of booting the Pi, accessing the GPIO pins and so forth.

One of the best is Baking Pi (**www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/index.html**) from the University of Cambridge. It's a detailed set of assembly language tutorials that step through turning on LEDs, accessing pixels on the TV, getting input from the keyboard and so forth. You have to learn a lot about the Pi's hardware on the way, and the tutorials were written for the original models of the Pi – so there's no guarantee that they'll work on the A+, B+ and Pi 2 models.

If you'd rather go the C route, Valvers at **http://tinyurl.com/qa2s9bg** explains the process of setting up a cross-compiler and building a rudimentary operating system kernel, while the ever-useful OSDev Wiki at **http://wiki.osdev.org/Raspberry_Pi_Bare_Bones** also shows you what you need to get a minimal kernel up and running.

As mentioned, the big problem is writing drivers for the various bits of hardware on the Pi: the USB ports, the SD card slot and so forth. The code for these alone can take up tens of thousands of lines. If you really want to make your own fully-fledged Pi operating system, it'd be worth going to the forums at **www.osdev.org** and asking if anyone else has written drivers for these bits of hardware – and maybe you can then adapt them for your own kernel, saving you heaps of time.



```
        .global _start
_start:
        ldr r1, =string1
        mov r2, #string1len
        bl print_string

loop:
        mov r7, #3      @ read
        mov r0, #0      @ stdin
        ldr r1, =char
        mov r2, #2      @ two chars
        swi 0

        ldr r1, =char
        ldrb r2, [r1]
        cmp r2, #113    @ ASCII for "q"
        beq done

        ldr r1, =string2
        mov r2, #string2len
        bl print_string
        b loop

done:
        mov r7, #1
        swi 0

print_string:
        mov r7, #4
        mov r0, #1
        swi 0
        bx lr
myfirst.s                                   18,0-1        Top
```

```
mike@okachi ~/asmtest $ as -o myfirst.o myfirst.s
 && ld -o myfirst myfirst.o
mike@okachi ~/asmtest $ ls -l myfirst
-rwxr-xr-x 1 mike mike 1.2K Nov 22 15:31 myfirst
mike@okachi ~/asmtest $ ./myfirst
Enter q to quit!
a
That wasn't q...
b
That wasn't q...
c
That wasn't q...
1
That wasn't q...
2
That wasn't q...
3
That wasn't q...
q
mike@okachi ~/asmtest $
```

Vim (above left) is a superb editor for hacking assembly – get the syntax highlighting file for ARM from **http://tinyurl.com/psdvjen**.

programming before, a register is an on-chip memory store. Most CPUs only have a handful of registers, compared to millions or billions of storage locations in RAM, but registers are much faster to use.) ARM registers are plentiful and general purpose: there are 16 in total, from **r0** to **r15**, and they don't carry weird historical baggage like in x86 where some registers can only be used for certain purposes at certain times.

So, **mov** is very much like its x86 equivalent, and note the hash mark next to the **4** to show that it's a number and not a memory address. Here we want to use the Linux kernel's **write** system call to print our string; to use system calls, we need to populate registers with the appropriate numbers before asking the kernel to do its job. The number of the system call should go in the **r7** register, and **write** is system call 4.

In the next **mov** instruction we place the file descriptor for where the 'Ciao' message should be output – eg the screen or a file – in **r0**. We're using **stdout**, which is 1. Next up, we need to place the location of the string we want to print in the **r1** register, using the **ldr** ('load register' instruction; note the equals sign here to show a location rather than a direct number. At the bottom of the source code in the data section we define this string as a sequence of ASCII characters. For this 'write' kernel system call, we also need to tell the kernel how long the string is, so we do that by placing the value of **stringlen** into the **r2** register. (This **stringlen** value is calculated by taking the end location of the string and subtracting that from the starting point.)

So, we've populated our registers with the relevant values, and now we're ready to hand control over to the Linux kernel. To do this, we use the **swi** instruction, which is short for 'software interrupt' and basically switches execution to a routine in the kernel (just like **int** in the x86 tutorial series). The kernel looks at the contents of the **r7** register, sees that it's 4 and says "Aha, the calling program wants to print a string". Then the kernel checks the contents of the other registers, does the printing work, and hands control back to us.

So we have "Ciao!" on the screen, and now we want to end the program cleanly. We do this by placing the number for the **exit** system call into the **r7** register and calling software interrupt number zero once more. And that's it – the kernel terminates our program and we land back at the command prompt.

PRO TIP

In assembly language, it's a good idea to use comments liberally. We haven't used many in the code samples here, to keep them short and sweet in the magazine (and also because we've provided detailed explanations). But if you write a complicated routine that seems obvious to you now, think how it may look if you forget about ARM assembly for a few months and come back to your code later. You might have forgotten all the awesome tricks and shortcuts you made, and the code looks like gobbledygook. So add plenty of comments, even if they seem a bit too obvious at the time!

Subroutines, loops, conditionals

Now that we know how to write, assemble and link a simple program, let's move on to something more intricate. The following program uses subroutines for printing a string (so we can re-use some code and don't have to set up registers manually each time). It has a main loop where a message is displayed until the user enters 'q'. Have a read through it to see what you can understand (or guess!) already, and then we'll go through it in detail. Note that **@** symbols denote comments in ARM assembly.

```
        .global _start
_start:
        ldr r1, =string1
        mov r2, #string1len
        bl print_string

loop:
        mov r7, #3                      @ read
        mov r0, #0                      @ stdin
        ldr r1, =char
        mov r2, #2                      @ two chars
        swi 0

        ldr r1, =char
        ldrb r2, [r1]
        cmp r2, #113            @ ASCII for 'q'
        beq done

        ldr r1, =string2
        mov r2, #string2len
        bl print_string
```

Reverse engineering

Sometimes it's useful to take a binary file and convert it back into assembly language. The results of this are usually not pretty, in that they're missing the human-readable label names and comments that you added to your source code, but it can be helpful to see what the assembler is doing with your code. To produce a disassembly of the **myfirst** binary, run:

```
objdump -d myfirst
```

This will disassemble the executable code section of the binary (and not the data bit, because that's ASCII text). You'll see that the instructions largely match what you entered. Disassemblers really come to life when you're trying to analyse the behaviour of a program to which you only have the binary – eg a virus, or simply a piece of proprietary software that you want to emulate. Always keep copyright concerns in mind though! Disassembling a binary and copying and pasting its code straight into your project is a bad idea – just use it for learning.

```
        b loop

done:
        mov r7, #1
        swi 0

print_string:
        mov r7, #4
        mov r0, #1
        swi 0
        bx lr

        .data
string1:
        .ascii "Enter q to quit!\n"
        string1len = . - string1
string2:
        .ascii "That wasn't q...\n"
        string2len = . - string2
char:
        .word 0
```

Here we start off by putting a string location and its length into the appropriate registers for the write system call – but then we jump to our **print_string** subroutine further down the code. To perform this jump we use the **bl** instruction ('branch and link'), which stores the current location in the code so that we can return to it later with the **bx** instruction. The **print_string** routine simply populates the other registers for the write call, as in the first program we wrote, before calling the kernel and then returning to the calling code with **bx**.

Back in that calling code, we next have a label called **loop** – we're going to jump back to this point in a moment. But first we want to use another kernel system call, **read** (number 3), to grab a character from the keyboard. So we place 3 in **r7**, and then zero (**stdin**) in **r0** because we want a character from the user's input and not a file.

Following this we place the location where we want to store the character in **r1** – in this case, the **char** location at the bottom of our data section. (We actually need a word, ie two characters, to store the data here, as the enter key input is stored as well. In assembly language, it's always important to be aware of data overflows – there are no high-level niceties to save your hide here!)

Back in the main code, we put 2 in **r2** for the two characters that we're going to store, and then call the kernel to perform the read operation. The user enters a character and hits Enter. Now we want to see what this character contains: we put its location (**char** in our data section) in the **r1** register, and then using the **ldrb** instruction we load a byte from the memory location pointed to by **r1**.

The square brackets here make it clear that it's the data stored inside the memory location that we're interested in – not the location itself. So **r2** now contains a single character from the **char** bit of our data section, and it's the character that the user entered. Our next job is to compare the contents of r2 with the letter 'q', which happens to be 113 in ASCII (see **www.asciichart.com**). So we use the **cmp** instruction to perform the comparison, and then 'branch if equal' to the **done** label if **r2** is 113. If not, we then go on to print our second string, then branch back to the start of the loop with the **b** instruction.

Finally, after the **done** label, we tell the kernel that we want to exit – just like in the first program. To run this program, just assemble and link it as per the instructions for the first one.

So, we've covered a lot in a short space here, but it's always best to learn by actually typing in and trying out code yourself. There's no better way to get familiar with a language than doing experiments by making changes and seeing what effects they have. You can now write simple ARM assembly programs with input, output, loops, comparisons and subroutines. If you had never dipped your toes into the imposing pond of assembly language before today, hopefully it has made you more confident with the language and it seems less of mystical art that only a few geeks dare to master.

Of course, that's just the tip of the iceberg when it comes to ARM assembly language. There's a lot more to explore, so if you'd like us to cover the language in more depth in future issues, get in touch! Meanwhile, another good resource for learning more about ARM coding on Linux can be found at http://tinyurl.com/nsgzq89. Happy hacking! LV

ARM chips started off in the Acorn Archimedes range of computers, but now utterly dominate the mobile space. (Image: **http://tinyurl.com/qy9p2l5**)

Now that we know how to write, assemble and link a simple program, let's move on to something more intricate

Mike Saunders still thinks the Z80 is better than the 6502. Flame on, BBC and C64 fans!

CODE NINJA: WRITE A GAME USING SVG

Turn your browser into a games console with scalable vector graphics.

BEN EVERARD

WHY DO THIS?

- Build cross-platform games
- Add animations and interactivity to your website
- Become an indie games superstar

Scalable Vector Graphics (SVG) is, as the name suggests, a vector graphics format. This means that the images include details of what lines, shapes and objects go into making the picture, rather than just details of what pixels should be which colour. One advantage of vector graphics is that as you make them larger, software can redraw the image at a higher resolution, so they don't lose quality (hence the scalable in the name). Another advantage is that because you know what the component parts of the image are, you can manipulate them. In this tutorial, we're going to manipulate SVGs using JavaScript and HTML to create a simple side-scrolling game.

You can interact with SVGs using plain JavaScript, but there are some differences between browsers, and it can be a bit fiddly. It's easier to use a library that provides a wrapper to make it more straightforward.

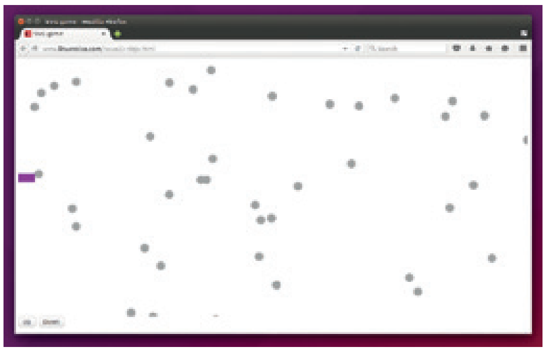
We're going to manipulate SVGs using JavaScript and HTML to create a simple side-scrolling game

We'll be using the **SVG.js** library.

The basic HTML for our game is:

```
<html><head>
<title>SVG game</title>
<script src="http://www.linuxvoice.com/svg.min.js"></
script>
</head>
<body>
<div id="drawing"></div>
<button onclick="ship.dy(-5)">Up</button>
<button onclick="ship.dy(5)">Down</button>
<script>
var draw = SVG('drawing').size(1200, 600)
</script>
```

In the HTML of this page, we've created a **div** element in which we'll place our SVG, and two buttons that we'll use to control the spaceship (the **onclick** code for these buttons links to code that we'll create later in the tutorial). The first script loads the **SVG.js** library from the Linux Voice server, and the second creates an SVG object that we'll manipulate to create our game. This SVG object has methods that enable


You can play our game online – live, right now! – at **www.linuxvoice.com/issue23-ninja.html**.

us to create elements of the SVG. For example, we can create a circle with a 10-pixel diameter with:

```
var circle = draw.circle(10)
```

Our game will be made up of asteroids (represented by circles) that move from right to left across the screen, and a spaceship (represented by a rectangle) that the user can move up or down. The aim is to keep the spaceship from colliding with the asteroids for as long as possible. The asteroids will gradually speed up, and your final score will be based on the speed the asteroids get to before a collision.

Asteroids!

The first part of our code sets up the variables we need, and places the asteroids at random positions across the SVG.

```
var draw = SVG('drawing').size(1200, 600)
var rocks = []
var num_rocks = 40
var rock_size = 20
var speed = 1000
var alive = true
for(var i=0;i<num_rocks;i++) {
  rocks.push(draw.circle(rock_size).fill('#aaaaaa' ))
  rocks[i].move(Math.random()*1200 + 200, Math.
random()*600)
}
```

This uses the array **rocks** to store details of all our asteroids. To create them in random positions, we first use the **circle** method to create the object, then use the **fill()** method to set them to the appropriate colour, and finally use the **move** method. **Math.**

random() returns a random number between 0 and 1, so by multiplying this by the dimensions of the image, we can generate random positions. To give the player a chance to get ready, we'll offest the initial x values by 200, so they're not immediately in front of the player.

The ship is represented by a rectangle, which is created by the **rect** method of our SVG object. This takes arguments for the width and height of the ship. The ship can't move forwards or backwards, just up and down, so we start by moving it to the middle of the left-hand edge of the screen. The **move** method places the element's top left-hand corner at the given coordinates, so this should have an **x** value of 0.

```
var ship = draw.rect(40, 20).fill('#aa00aa' )
ship.move(0,300)
loop_forever()
```

After setting the ship up, everything is ready for our game. We only need to start the main loop running. We do this by calling the function **loop_forever()** that will keep running until the game is finished. In most other languages, we'd use a **while** loop, but JavaScript doesn't have any way of pausing the code, so you can't easily control the speed of your game using plain loops. Instead, we'll recursively call the **loop_forever()** function using the **setTimeout** function to run the animation every 20 milliseconds:

```
function loop_forever() {
  //animate rocks
  for(var i=0;i<num_rocks;i++) {
    rocks[i].dx(-1 * speed/1000)
    if (rocks[i].x() < 1) rocks[i].move(1200,Math.
random()*600)
    if(rocks[i].x() < 40 && rocks[i].y() > ship.y() - 20 &&
rocks[i].y() < ship.y() + 20) {
      alert("Score: " + speed + ". Reload the page to try
again")
      alive = false
    }
  }
  speed++
  if (alive) setTimeout(loop_forever, 20)
}
```

The main action in this function is in the **for** loop, which loops through every asteroid and does three things. First, it moves all the asteroids left by the amount determined by speed/1000. The speed variable increases by 1 every time this loop is run, so this will gradually speed up as the game runs. The **dx()** method moves an SVG element relative to its current position. There's also a **dy()** method to move an element vertically which we tied to the click events on our buttons in the HTML.

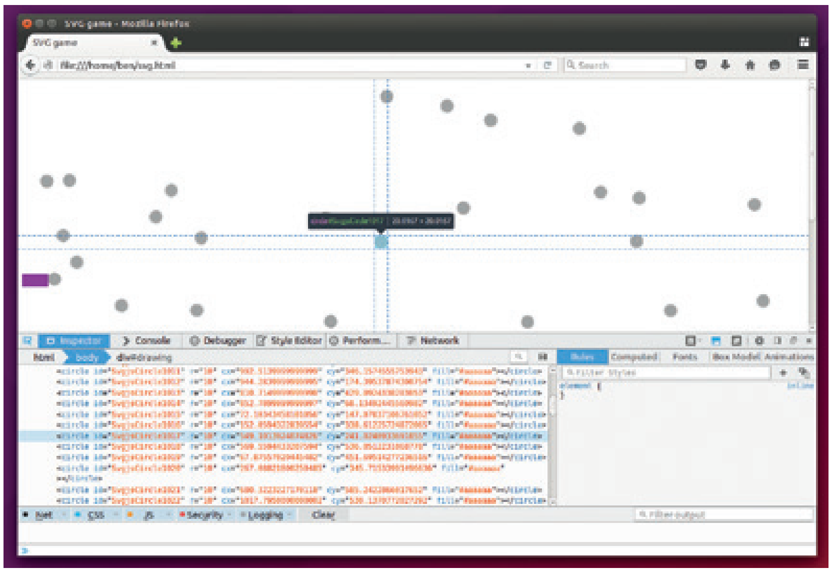
Second, the first **if** statement checks for any asteroids that have moved off the left-hand edge of the screen and places them at a random position on the right-hand edge of the SVG. The **x()** method returns the current **x** position of an element. There's also a **y()** method that does the same for the vertical, which we'll use a bit later. You can use either of these with a single argument to set the horizontal or vertical position, but we don't need this in our game.

Collision detection

Third, the final **if** statement does a simple collision check. This isn't perfect, since it checks the bounding rectangle of the asteroid against the ship, but this is accurate enough for our needs. **SVG.js** doesn't have any collision-detection methods built in, so if you want to increase the accuracy of this, you'll have to write the code yourself.

If there is a collision, the game displays an alert to let you know your score, and sets the **alive** variable to **false** to stop the game loop continuing. At this point, you could reset the game, but in the interests of saving space, we've just asked the player to reload the page to continue.

And that's it: a game in under 30 lines of JavaScript. If you want to extend it, there are loads of options, such as improving the graphics (you can load images in the same way we created shapes using the **draw.image(url)** method), changing the difficulty level or adding features such as a gun on the ship.



You can use the *Firefox* inspector (Ctrl+Shift+I) to see elements inside the SVG, which can help debug any problems.

SVG vs Canvas

As well as SVG elements, you can create Canvas elements in HTML which also hold scriptable graphics. The fundamental difference between the two is that SVGs contain the elements that make up the image (such as the circles and rectangles that we drew) while canvases just allow you to draw bitmap images. They don't hold the elements that go together to make up these images meaning that you, as the programmer, have to do more work to keep track of what goes where. The pay off for this extra work is that you can get higher performance out of a canvas, particularly if there are lots of bitmaps displayed.

Ben Everard is the best-selling co-author of the best-selling *Learning Python With Raspberry Pi*.

MINSKY PART 2: ECONOMIC MODELLING

Master global wealth fluctuations and understand every word in the FT...

ANDREW CONWAY

WHY DO THIS?

- Correct the pub bore who oversimplifies (and misunderstands) the news
- Make money in your pyjamas on pork belly and orange juice futures
- Play God with an economy despite only having a fragile understanding yourself!

The financial crisis of 2008, which saw banks fail or survive only with lifelines thrown to them by governments, prompted many to take a greater interest in economics and ask why the "experts" failed to see it coming. *Minsky* is free and open source software that can model many types of dynamic systems, but its *raison d'être* is to model the economy: the flows of money through businesses, consumers, government and banks.

Minsky was created by Steve Keen, one of the few economists that did see the 2008 crisis coming. He named the software in honour of economist Hyman Minsky, who argued that economies were inherently unstable and prone to sudden crashes, known as "Minsky moments".

We're going to use *Minsky* to enter the world of economic modelling. If you didn't see part 1 of this tutorial in issue 22, do take a moment just now to read the Quick Start boxout. All the screenshots produced for this article have a corresponding **.mky** file you can load up for yourself, though we recommend you try and wire up at least the most basic models. You can grab the **.mky** files from **https://github.com/mcnalu/linuxvoice-minsky** and load them using the Open item under the File menu.

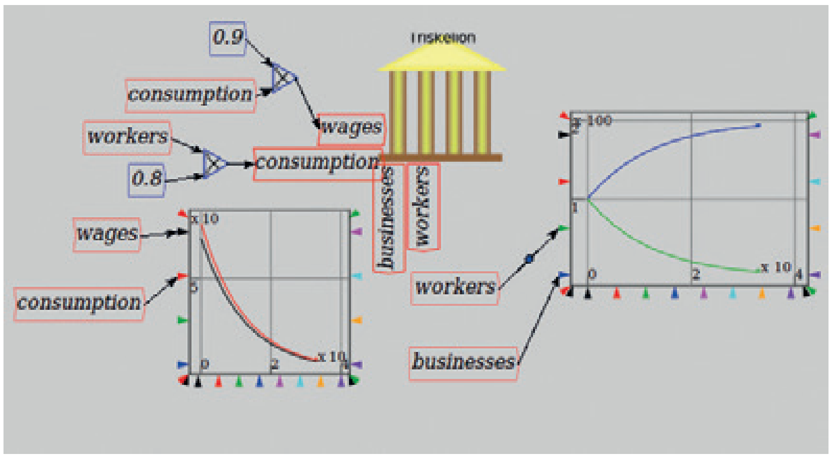
Consume what you earn

Let's model a very simple economy of the planet Triskelion, which starts with no banks nor a government. Imagine that there is full employment, so everyone is a worker who works for a business. We're only going to track two quantities: the money held by all businesses, and the money held by all workers. Let's suppose both totals start (at **t=0**) at 100 Quatloos or Q100, the Quatloo being the currency of Triskelion. Also, the total wage bill to pay all workers is constant at Q100 per Triskelion year. One year is equivalent to **t** changing by 1 in the simulation (see the top-right of Minsky's window). Let's also suppose that the good people of Triskelion always spend 80% of their wealth in each time period.

The boxout over the page shows how to set up this simple model in *Minsky* using the Godley Table, which uses double-entry bookkeeping to ensure that our flows of money are correct, and we're not leaking or creating Quatloos unintentionally. When you run this model you'll see that workers' wealth rises from Q100 and levels out at Q125 after five years or so, whereas the business total drops from Q100 down to Q75. As we hoped, the number of Quatloos in the economy remains constant at Q200.

Now, you might have thought the Triskelion people were a bit free with their money, always spending 80% of whatever they had available. If so, you might be surprised to see that the workers did well out of this: overall, Q25 went from businesses to the workers. You can see why this balance occurred, because once achieved the workers spend 80% of Q125, which is Q100 and equal to their wages. What this example illustrates is that consumers are limited to spending what they get paid, and that in turn determines profits (or losses) of businesses.

In the paradox of thrift both businesses and workers try to save money and cause a disaster.



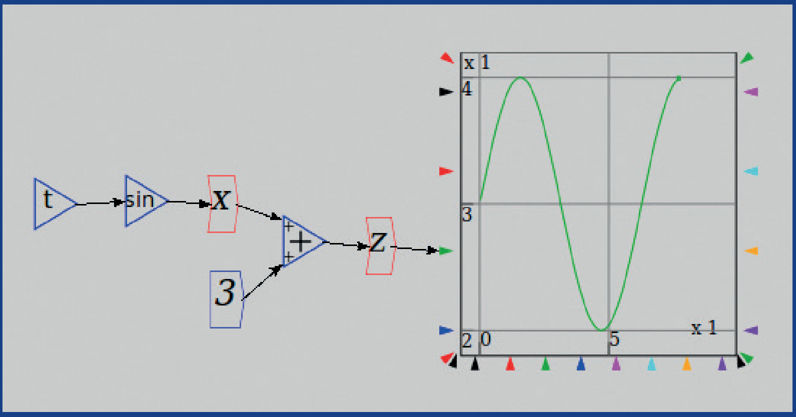
The paradox of thrift

One of the business owners notices that workers are saving 20% of their wages and that the business is currently making a loss. The owner might well conclude that she needs to cut the wage bill. She decides to replace the fixed Q100 wage bill with one that's set at 90% or 0.9 times consumption. This means the simulation starts with consumption at Q80 and wages at Q72, so businesses make a profit of Q8 per year. When you run this simulation you'll see that business ends up with all the money and the workers with none. But look at the graph of wages and consumption – this is not a good outcome for businesses, but a disaster all round! Workers end up

Minsky quick start

You can build *Minsky* from source, but we used the binaries available from the OpenSUSE build service – see **sourceforge.net/projects/minsky**. In *Minsky* you are building a numerical simulation using components and wires rather than lines of code. Components either have a value, such **const** (a constant) and **var** (a variable), or are operators, represented by a triangle pointing to the right with a symbol inside. All of these can be placed by clicking on the symbols on the toolbar and clicking again to place them on the canvas.

For example, to calculate z=x+3, you would place a **var** called **x**, another **var** called **z** and a **const** set to **3**. If you set **x** to 5, either on creation or via a right-click, then **z** will take on the value of 8. However, this is dull because it doesn't change with time. To make things more interesting, drop in the special **t** operator (which generates time), then **sin** and wire it all up as shown on the left of the screenshot. Finally, place a graph using its icon (bottom-right on the toolbar) and wire **z** up to a port at its left. Now hit the square button in the toolbar to reset the simulation and hit the Play button to run the simulation and watch your graph of **z=sin(t)+3** be plotted against **t**. We won't be using the time operator directly in this tutorial because it's part of the Godley Table, which is key to *Minsky*'s economic modelling capability. As noted in the previous tutorial, *Minsky* is a work in progress, so a few quirks and bugs are to be expected.



Minsky is about economics, but you'll also learn a nice bit of maths while you're modelling.

with nothing, so spend nothing, and the businesses get no income and can only sit on their amassed Q200. This is known as the paradox of thrift, because if everyone tries to save at the same time (business is effectively saving by retaining profits) then it leads to an economic dead-end.

In reality, the disastrous end-point is not reached, because workers will not be able to save once their income has fallen so low that they cannot afford basics such as food and housing. However, real societies can end up with significant problems arising from increasing inequalities of wealth and income.

Government – spend and tax

Eventually Triskelion introduces a government, mainly to help manage economic collapses like the one we've seen, but also to help regulate an endemic gambling problem. Government becomes a new stock column in the Godley Table, and two new flows are introduced. There is now an income tax of 20% or 0.2 on worker's wages that goes to the government, but the government also spends Q25 per year on paying its own workers, such as the teachers and doctors. These lucky government employees on Triskelion pay no tax.

You'll notice that the wealth of the government – shown by the black line on the graph – is negative. It starts at zero and decreases because the government spends Q25 but only gets back Q20 per year in taxes (20% of Q100). So, overall, money is flowing out of the government – the public sector – and to the workers and businesses – the private sector. At first the workers benefit most, but as their wealth rises so too does their consumption until it equals their wages. At this point the workers' wealth reaches a maximum and the money flowing out of government goes via consumption to the businesses, whose wealth continues to rise.

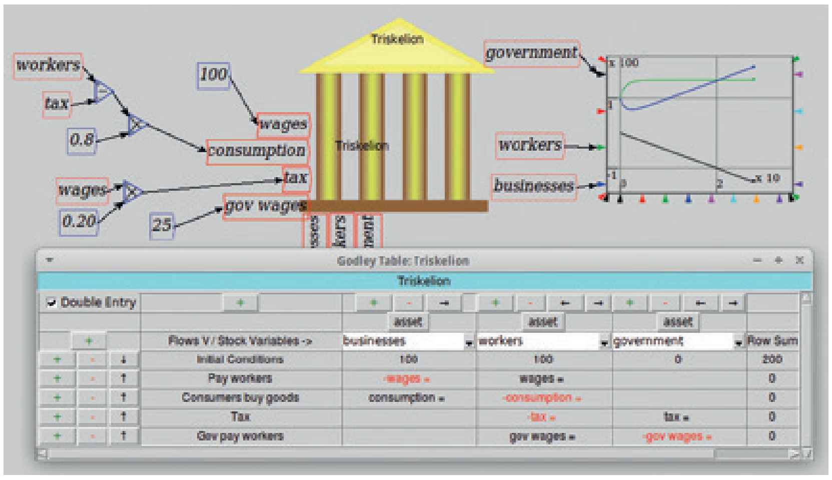
If you check, you'll find that overall wealth – adding up government, workers and businesses on the graph – is constant at Q200, as before. But in time businesses will accumulate much more than Q200 in wealth. You can interpret the negative wealth of government in two ways: either the Triskelion government is becoming indebted to another planet,

> If everyone tries to save at the same time (businesses do this by retaining profits) it leads to an economic dead-end

or they are simply creating Quatloos out of nothing (being the government, they can do that).

After 30 years, the government decides to stem this outpouring of Quatloos by raising the tax rate to 25% so that government taxes equals its spending (both Q25). Investigate this for yourself: pause the simulation at t=30, edit the 0.20 under wages to 0.25 and unpause.

Now a government spends money on paying government employees, but taxes workers employed by businesses.



Godley Table: Triskelion

Triskelion

☑ Double Entry

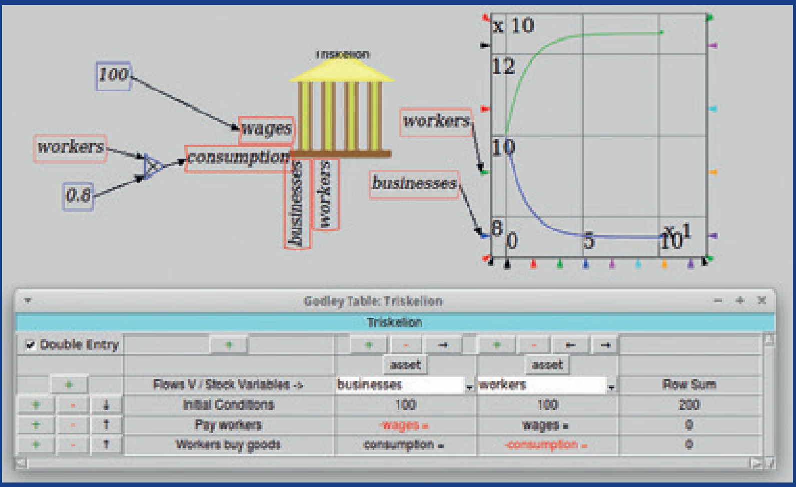
	asset	asset	asset	
Flows V / Stock Variables ->	businesses	workers	government	Row Sum
Initial Conditions	100	100	0	200
Pay workers	-wages =	wages =		0
Consumers buy goods	consumption =	-consumption =		0
Tax		-tax =	tax =	0
Gov pay workers		gov wages =	-gov wages =	0

The Godley Table

Let's demonstrate how to use the Godley Table with a simple model of Triskelion's economy. First, click the orange icon to place the Godley Table (named in honour of economist Wynne Godley, who among other thigs was also a director of the Royal Opera) on the canvas.

Now double-click on the placed icon and in the window that opens click on the button that says **noAssetClass** and select **asset**. Then click in the white area immediately beneath and enter **businesses**, and click the **+** button at the top to create a new column, make it an asset and label it **workers**. In the row below that says **Initial Conditions**, set both to 100, then click the **+** button on the left to create a new row. Click in the first column and name the new row **Pay workers**, then under **workers** enter **wages** and under **businesses** enter **-wages**. This creates a variable that means the business total will be reduced by that amount and the workers' increased by an equal amount. Now in the same way create the row called **Workers buy goods**. Workers spend an amount called **consumption**, which goes from the workers to the businesses.

Notice that the **Row Sum** column is zero for these two rows. This is an example of what's called double-entry bookkeeping, where the same amount is shown as a credit (positive) and a debit (negative).



Godley Table: Triskelion

Triskelion

☑ Double Entry

		asset	asset	
	Flows V / Stock Variables ->	businesses	workers	Row Sum
+ - ↓	Initial Conditions	100	100	200
+ - ↑	Pay workers	-wages =	wages =	0
+ - ↑	Workers buy goods	consumption =	-consumption =	0

The variables **workers** and **businesses** shown at the bottom of the Godley Table icon are stocks because they represent an accumulated stock of money, whereas wages and consumptions are flows, because they move between stocks each year.

Minsky shows the Godley Table as a columned portico that resembles the frontage of the Bank of England

The Triskelion government does a good job of managing the economy for many years, but then succumbs to populism and gears the entire economy around betting on gladiatorial combat. Fortunately, a starship visits the planet and its libidinous captain deposes the government. However, following his departure the economy enters turmoil in which both loss-making businesses and cash-strapped workers panic, try to save and via the paradox of thrift drive the economy into a deep recession; wages fall and get stuck at Q50 per year.

The situation is modelled in the final boxout with the workers having zero wealth and the businesses having Q200 at t=0. Wages of Q50 per year are well below the Q70 per year that the workers need to survive, eg to buy food and pay rent. The businesses, seeing their future is in peril too, decide to offer the workers credit, lending them the difference between their wages and the consumption they need to survive. In return, the businesses expect interest to be paid at 10% per year. This is represented in the Godley Table with a new stock called **workers debt** and new flows called **lend** and **interest**, which are defined to the left of the Godley Table icon.

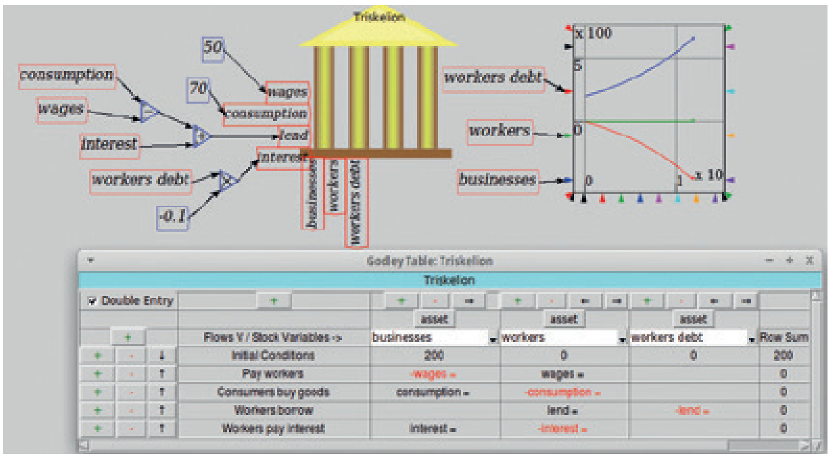
You may have spotted the problem before even running the simulation: the workers will need to borrow the Q20 shortfall every year, and in addition they'll need to borrow the money needed to pay the interest. As such, their debt increases exponentially as does the assets of the business: one person's debt is another's asset. This means great inequality develops, with the businesses holding all the wealth and debt assets, and workers with no wealth and ballooning debt liabilities. Although we've not had space to cover it, the Godley Table can be set up to represent assets, liabilities and also equity.

Apologies, congratulations and where to go

We are sorry to say we have tricked you into creating and solving differential equations, and for that congratulations are also due. If you don't believe us, then click the **equations** tab at the top-left of the screen to see them. If you like *Latex* and want to print out prettier equations (say, to frame and put on your wall) use the **Output Latex** item under the File menu.

You may well have noticed that these hypothetical situations on Triskelion have some parallels with societies on our own planet. If you'd like to learn more, feel free to experiment with your own models and also to delve into the more complex models explained in Steve Keen's video tutorials on his website: **http://www.debtdeflation.com/blogs/minsky**.

Andrew Conway aspires to be a libidinous astro-economist starship captain, bestriding the universe like a colossus.



Godley Table: Triskelion

Triskelion

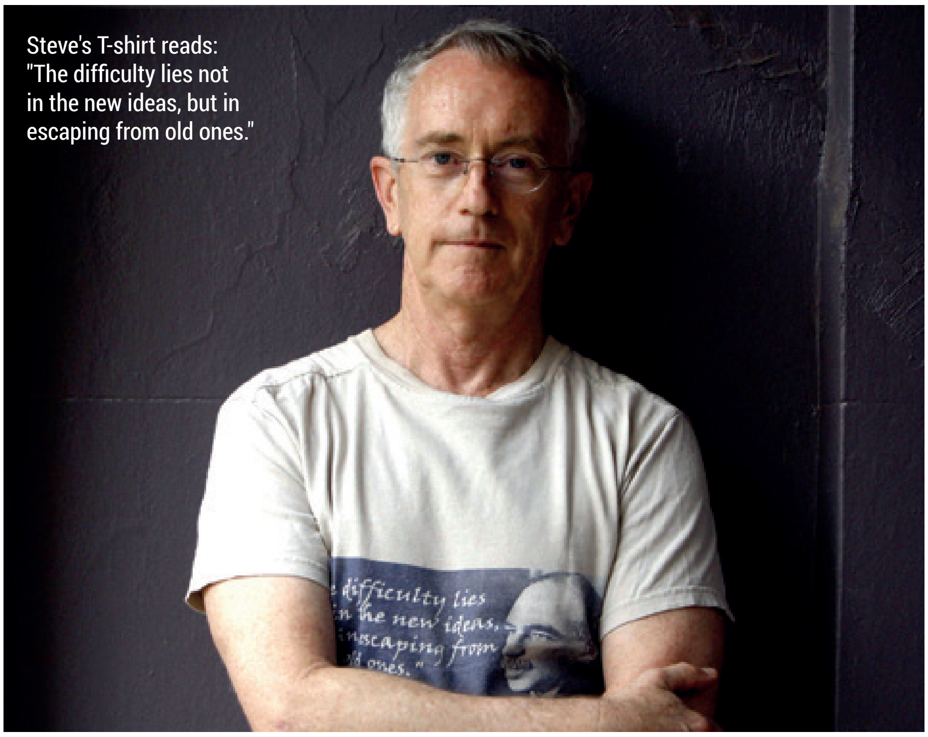
☑ Double Entry

		asset	asset	asset	
	Flows V / Stock Variables ->	businesses	workers	workers debt	Row Sum
+ - ↓	Initial Conditions	200	0	0	200
+ - ↑	Pay workers	-wages =	wages =		0
+ - ↑	Consumers buy goods	consumption =	-consumption =		0
+ - ↑	Workers borrow		lend =	-lend =	0
+ - ↑	Workers pay interest	interest =	-interest =		0

Now a government spends money on paying government employees, but taxes workers employed by businesses.

INTERVIEW STEVE KEEN

Minsky is the brainchild of Professor Steve Keen who is currently Head of the School of Economics, History and Politics at Kingston University in London. We caught up with Steve just before he headed off to the House of Lords to offer advice on economics.



Steve's T-shirt reads: "The difficulty lies not in the new ideas, but in escaping from old ones."

LV What led you to create *Minsky*?
Steve Keen: I've always been a critic of mainstream economics. It makes several fundamental assumptions that I oppose, but particularly that the economy is in equilibrium or is heading towards it, and that it can be modelled without describing how banks handle money.

I also have a background in computing and was editor of software magazines in Australia from the early 1980s to mid-1990s. During my PhD I used *MathCAD* and *Mathematica* for non-linear modelling and also came across *VisSim*.

I started out modelling the role of money in the economy taking inspiration from a predator–prey model developed by mathematician John Blatt, added debt-based investment to it and ended up with a chaotic model [related to the Lorenz model discussed in part 1]. That work was in 1992 and since then I've been working to include banking in the model. At first I worked with differential equations by hand, then used a matrix approach with *MathCAD*, which in turn led to the tabular format used in *Minsky*.

LV So that's what led to *Minsky*'s unique Godley Table feature?
SK: Yes. I knew I wanted to design a software package around the idea and obtained a $128,000 grant from the Institute of New Economic Thinking (INET). At this point I brought in Russell Standish, who's currently a visiting professor at the University of New South Wales, who has a background in physics and who'd already built the *Ecolab* software (a dependency of *Minsky* also available on SourceForge). Russell then wrote the code for *Minsky*, so I found myself not only working with a good programmer but a good friend who understood my approach to economics.

LV Whose idea was it to release *Minsky* under the GNU GPL?
SK: Russell works with open source software and I preferred it because I knew it'd help with adoption. Interestingly enough, when I spoke to the then director of INET he said it's got to be open source otherwise people will think George Soros (co-founder of INET) is trying to make money out of it!

LV *Minsky*'s visual approach has the potential to get folk to engage with new economic ideas without requiring them to have a conventional mathematical or economics training.
SK: Exactly. To engage people you need to be visual. Economics is still stuck in a tedious world of intersecting lines and algebra with differential equations hardly getting a mention. A good way to encourage people into my approach of non-linear and non-equilibrium thinking was a visual tool.

LV *Minsky* has been updated in the last year, but how much time has Russell got to work on it?
SK: He's got to earn a living from coding so needs funding to work on *Minsky*. In 2013 we raised $78,000 from 620 backers on Kickstarter and more recently I've put some of my own money into it, some which I've earned by giving talks to banks for a sizeable professional fee. People can also give money via my blog **www.debtdeflation.com/blogs/minsky**.

LV Is there other free and open source software that is similar to Minsky?
SK: There's *Xcos*, which is part of *Scilab* – a clone of *MathWorks Simulink*. There's also *Modelica* which, although a compiler, has various graphical front-ends. We thought about basing *Minsky* on *Xcos*, but found it too cumbersome. Had we known about *Modelica* at the time we might have used it as the basis of *Minsky* rather than coding from scratch in C++.

LV Where do you see room for improvement?
SK: On the economics side, I'd like to add multi-sectoral modelling (eg private and public sector or multiple countries). I've done that by hand: a few years ago I was asked by the UN environment programme to provide a non-equilibrium, multi-sector economic model to provide input to an ecological model of the environmental impact on southeast Asia.

LV What would you do if you had a million-dollar budget for Minsky?
SK: Go multi-user so I can take policy-makers through the economic model where they can see the effects of varying taxation and spending. That way I could demonstrate to them that if they do simplistic things like reducing the government deficit they risk increasing the instability in the system by driving people to run up higher private debt, which caused the last financial crisis. I'd also like to create a model repository so you can have an HTML 5 application with several users building a model together. Ultimately, the goal is to properly model capitalism. LV

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.

Privilege separation

Typical Linux system hosts multiple users running many processes. How does it ensure all these inhabitants play safely?

If you've ever tried to prove to someone that Linux is better than Windows, you'll have used security as one of your arguments. Regardless of the improvements that Windows has made in this field in recent years, Linux still enjoys the reputation of more robust, solid operating system. A well-thought out process privileges mechanism is what earned Linux this reputation. You can't do any harm to your system unless you're logged in as root, and if you use Linux properly, you are almost never logged in as root. Today we'll see how these permission bits stick together to ensure safe operation of our desktops, servers, and even mobile devices.

A two-numbers game

At the centre of the Linux privilege system lie two integers: these are User ID, or UID, and Group ID, or GID. They may come in different "flavours", but ultimately these are the credentials that the kernel uses to determine if a process should be granted access to some resource, like a file, or rejected. A UID and GID of zero are reserved for root, and normal user accounts often start at 1000, but the latter is merely a convention.

People aren't good at remembering numbers, so UIDs and GIDs are usually assigned human-readable names. This is similar to how DNS maps hostnames to IPv4 (or IPv6) addresses. Most often, UID and GID mappings are stored in **/etc/passwd** and **/etc/group**, respectively. You can read them with the **getent** command:

```
$ getent passwd val
val:x:1000:1000::/home/val:/bin/bash
```

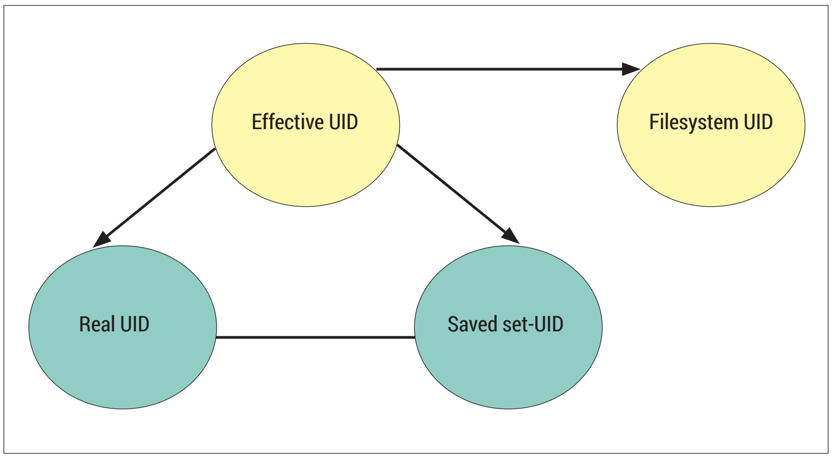
To do this in the code, refer to **getpwnam(3)** and related functions. UID is the number in the third column, and the next one is the primary group's GID. They don't have to be equal, but often coincide, as Linux system designers tend to put each user in a separate group.

Any user in Linux may also belong to some supplementary groups. You can change this membership via the **usermod -a -G** command; note that this requires root privileges. Supplementary groups are convenient for rights management: say, you add users to the **plugdev** group to let them access external storage devices in Ubuntu.

Despite the text above attributing credentials to human users, they are really associated with processes. In fact, processes may have several instances of both UIDs and GIDs. The kernel switches them under strict rules, and this is how privileges change in Linux. Let's go through this step by step. We'll speak of user identifiers; for group identifiers, things stay precisely the same.

The three "flavours" of UID are 'real', 'effective' and 'saved'. A real UID determines who owns the process. The effective UID is the value that the kernel checks when evaluating resource access requests. The process inherits both from its parent across **fork()**, and usually they are equal to each other.

A third flavour, saved set-user-ID, comes into play with SUID binaries. Ordinary programs have the privileges of the user who executes them. Specific programs, like **passwd**, may need higher privileges, however. Such programs often have a SUID bit set



For unprivileged processes, IDs aren't assigned but "rotated" between values already in the set. A change in effective ID updates filesystem ID as well.

with **chmod +s** (usually, during packaging). When the kernel starts a SUID program via **execve()**, it overrides the effective UID and saved set-user-ID with the file owner's UID (often 0).

For historic reasons, Linux also defines filesystem UID. The kernel automatically keeps filesystem UID in sync with the effective UID, unless you change the former manually. We won't, and this Core Tech won't deal with filesystem UIDs.

Why bother to have so many identifiers for a single process? The trick is how Linux switches between them. A privileged process with an effective UID of **0** can use **setuid()** and related system calls (see the boxout) to change the IDs arbitrarily. A non-privileged user, however, is restricted to values that are already in the (real, effective, saved) set. So it can, say, change effective UID to real UID or saved set-UID, but can't set it to **0** and become root.

For instance, consider a program that starts as root but then drops privileges with **seteuid(1000)**. When it decides it wants to make a privileged operation once again, it can issue **setuid(0)** and this will work, because zero matches its real UID. This ability to escalate once-dropped privileges back could be a security breach. To prevent this behaviour, most programs use **setuid(1000)**, which updates all three UIDs if the caller effective UID is zero.

Set-UID programs work in a similar fashion. Consider a SUID program owned by a non-root user, but not you. You run it, and at some point it decides it doesn't need owner privileges anymore. So, it uses **setuid(getuid())** to set an effective UID equal to the real one (ie, "become you"). However, it can switch back to its owner's UID at any time, as the latter is also in the set as a saved set-UID. Unlike a root-owned program, an unprivileged one can never forget its origins. The reason for this is that it can't set any UIDs to the value not in the (real UID, effective UID, saved set-UID) set.

Gaining privileges

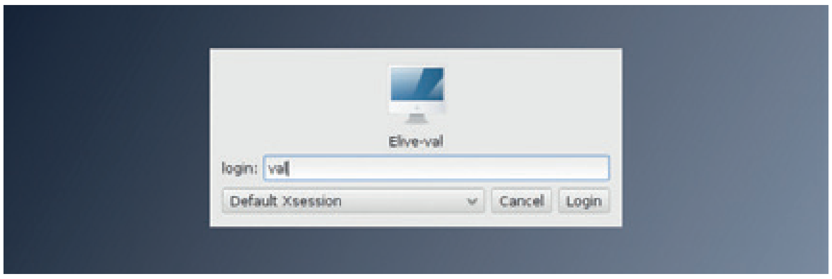
Now when we know how privilege mechanics works in Linux, let's see how processes obtain their privileges. LV019 already covered daemons, so we'll stick to interactive processes here. For these, privilege assignment occurs mainly in two spots: when we log in, or when we launch something with **su**/**sudo**.

For most desktop Linux distributions, login happens via display managers, like *KDM* (KDE) or *GDM* (Gnome). Both are relatively complex pieces of software. So, for this Core Tech, we'd opt for classical and simpler **login**, which manages text-based terminal sessions.

The **login** command comes as a part of the **util-linux** package, and you can find it as **login-utils/login.c** in the sources. The actual authentication happens in PAM (Pluggable Authentication Modules), which is not our focus today. If all goes well, the program calls **getpwnam()** to get a password database entry (**struct passwd**) for the user whose name was passed to **login** as a command line argument. If the authenticating user is not root, **login**

The kernel switches UIDs and GIDs under strict rules, and this is how privileges change under Linux

calls **initgroups()** to fill the supplementary group list from **/etc/group** (for root, it's empty). Then **setgid()** is called to set the primary group ID. As **login** runs as root, any value is acceptable, and all three group IDs are updated. From now on, the process runs with



login: val

Enlightenment's display manager can also set process credentials, and does this with a jolly dash of eye candy.

Managing user and group IDs

Linux provides quite a few system calls and library functions to get or set user and group IDs.

User ID	Group ID	Operation remarks
getuid()	getgid()	Return real user or group ID of the calling process.
setuid()	setgid()	Set all three IDs (superuser) or just effective ID (non-privileged).
geteuid()	getgid()	Get effective IDs of the calling process.
seteuid()	setegid()	Set effective IDs. Doesn't affect other IDs (e.g. real) for privileged processes.
getresuid()	getresgid()	Get real, effective and saved set-ID of the calling process.
setresuid()	setresgid()	Set real, effective and saved set-ID (superuser) or "rotates" them (non-privileged).
getfsuid()	getfsgid()	Get filesystem ID (defaults to effective ID).
setfsuid()	setfsgid()	Set filesystem ID (reset to effective ID whenever the latter changes).

Besides, Linux provides **getgroups()**/**setgroups()** system calls to get or set a supplementary group list, and the **initgroups()** function, which initialises the supplementary group list as per **/etc/group**.

popa3d – a tiny (yet real-world) POP3 daemon from the producers of *John the Ripper*.

authenticating user group privileges, but it still has an effective UID of root.

Then, **login** sets environment variables (like **$USER** and **$HOME**) from the **passwd** entry, prints **motd** and forks to create a new session (see LV019). Recall that group IDs (and the environment) are inherited across forks. The parent watches the child, which does **setuid()** shortly afterwards. As it still runs as root, this updates all three user IDs. From now, the process is wholly owned by the authenticating user. Unless that user is root, it's non-privileged, and has no way to switch back to root (which would be a security hole). The only thing left for **login** is to **chdir()** into the home directory and run the shell.

When I said there is no way to switch back to root, I lied a bit. We all know how to regain root privileges with **su** or **sudo**. Both are separate root-owned programs (so technically I was correct) and both have set-UID bit set. **su** is also part of **util-linux**, and is more flexible as it can switch to any valid user, not only root. "su" means "switch user", while "sudo" is "do as super user".

PRO TIP

You can find a really short, pseudo-code like **su** implementation in the Android sources: **https://android.googlesource.com/platform/system/extras/+/master/su/su.c**.

When **su** starts, the kernel sets the process effective UID and saved set-UID to **0**. Then **su** does most of what we already saw in **login**. It reads a password database entry for the new user (root, by default) and performs authentication via PAM. Then it initialises supplementary groups, and calls **setgid()** and **setuid()** for the new user. The order is important: if **su** did it the other way around, the **setgid()** call could fail, as the new group ID doesn't match your effective GID, real GID or saved set-GID. I say "your", not "root", because **su** is set-UID, not set-GID. After that, **su** modifies the environment and, optionally, creates a new session. Finally, a user-supplied command or a shell is executed with **execvp()**.

These are not the only ways processes gain their privileges, but the idea stays pretty the same in all cases. **su** and **sudo** are preferred ways for short-lasting administrative tasks like installing updates – you shouldn't log in as root, you know. This all works great for end-users, but what if you need a non-interactive daemon to perform some selected privileged operations in a safe way? What you are looking for is privilege separation, and there are some well-known ways to implement it in Linux.

Playing safely

Daemons usually start as root. Most often, this happens from an init script running as root, and daemons simply inherit their parent's permissions. Even if you start a daemon from the terminal, you have to use **sudo**, as it is considered a privileged operation. Furthermore, daemons often need superuser privileges, for instance, to bind network ports below 1024.

However, most of the work daemons do is perfectly bearable for a non-privileged user. So, as the principle of least privilege stipulates, they should drop root privileges as early as possible. Calling **setgid()** followed by **setuid()** will do the trick, but what if the daemon wants to regain superuser rights occasionally?

Consider a POP3 mail server. Once bound to a port (110), it can accept connections, handle the protocol and read mailboxes as non-privileged user, provided that mailbox files have the appropriate permissions. However, it also needs to authenticate users. Depending on the password database used, this may imply being root. For instance, it'll need superuser privileges to read **/etc/shadow**. Truth be told, today's mail servers are usually not POP3 but IMAP. Furthermore, they don't use system-wide password databases, but this is how real mail servers worked about 10 years ago.

popa3d (**www.openwall.com/popa3d**) is one such mail server. It was built for the Openwall project

Capabilities

We got used to the concept of almighty root user, but it doesn't have to be so. Instead of the brute all-or-nothing approach, you can assign granular privileges to specific programs or execution threads (think processes).

In fact, the kernel doesn't check that you are root. Instead, it evaluates process capabilities, as in this excerpt from the **setuid()** system call handler:

```
if (capable(CAP_SETUID)) {
    new->suid = new->uid = uid;
    ...
```

There is a set of dedicated system calls to manage per-thread capabilities. For details on using this mechanism, refer to **capabilities(7)**. Alternatively, there is the **libcap** (not to be confused with the **libpcap**) package, which provides a higher level, more stable process capabilities interface. It also includes two tools – **setcap** and **getcap** – to manage file capabilities. Chances are, you already use file capabilities without even noticing it:

```
$ getcap /usr/bin/ping
/usr/bin/ping = cap_net_raw+ep
```

ping creates raw network sockets, which is a privileged operation. So, my Arch Linux system grants it **CAP_NET_RAW** capability, and I can execute **ping** as a non-privileged user. Alternatively, **ping** could be made into a SUID binary, which is the case for Ubuntu:

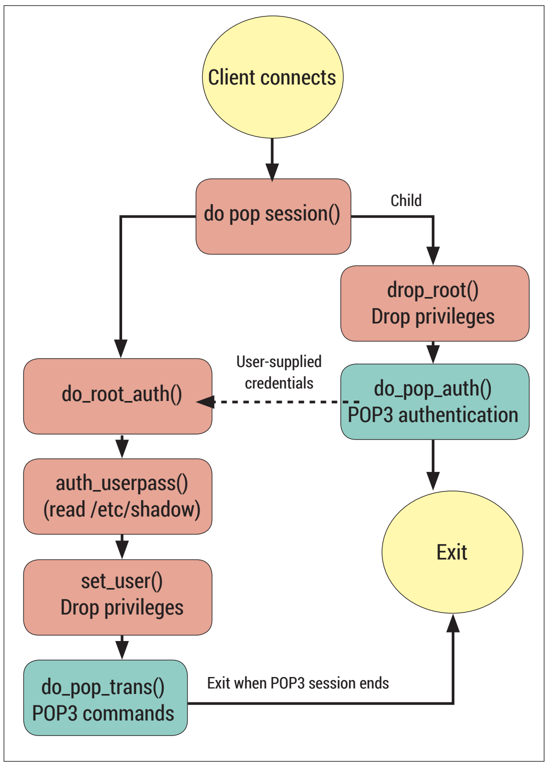
```
$ ls -l /bin/ping
-rwsr-xr-x 1 root root 44168 May  8  2014 /bin/ping
```

File capabilities require filesystem extended attributes support, but it pays with greater overall security.

with security being the top priority. It's relatively short (about 3500 SLOC) and currently comes as a standard in Slackware.

So, how does *popa3d* separates privileges? Let's track it from the entry point: the **do_pop_session()** function defined in **pop_root.c**. First, it creates an unnamed pipe and stores it in the **channel** variable. This pipe will be used later as a simple IPC mechanism. Then the daemon forks, **chroot**s to **/var/empty**` (see below) and drops privileges. From now, two processes are handling the session: the parent and the child. The parent runs as root and spins waiting for data in **do_root_auth()**. The child is unprivileged, and it runs the **do_pop_auth()** function which eventually supplies client-provided credentials into **channel[1]**. Then the privileged parent reads them from **channel[0]** and checks them against **/etc/shadow**. After that, it doesn't need root permissions anymore, so it drops privileges again and continues handling POP3 protocol messages.

So, *popa3d* has both privileged and non-privileged processes, which communicate over an unnamed pipe. This is quite a common pattern, and the benefit is that the privileged process is never exposed directly to the outside world. This way, if a service appears to be remotely exploitable, the damage would hopefully be minimal. Privileged operations are requested through the IPC channel as needed. However, one can't use this mechanism to run arbitrary commands. In the example above, one can check a username–password pair, but you can't order a system shutdown.



popa3d processes communicate over an unnamed pipe to achieve proper privilege separation. Red blocks are code that runs as root.

Escaping the jail

It's important to understand that there is no such thing as ultimate security. Imagine the **do_root_auth()** function contained a bug that caused buffer overflows on long usernames. A remote attacker could still trigger the vulnerability, even if he never interfaces with the privileged process directly. A false sense of security is worse than no security at all, and one mechanism that's often overestimated is the **chroot()** system call. You know that in Linux, the filesystem has a single root. This means, any file or directory in the system is ultimately a child of **/**. In fact, this is a per-process setting, but most of the time all processes share the same view of a filesystem. In a nutshell, **chroot()** is a way to move a filesystem root directory down the tree. If a process does, say, **chroot("/var/empty")**, its view of a filesystem will be restricted by what's in **/var/empty**.

One can readily see this as a hardening mean. If a process has no access to data files, binaries or libraries, it can't do any harm to them, even if exploited. In fact, it may even refuse to run if a **chroot** misses some essentials like **glibc**, but that's a whole another story. It would in fact be a hardening mean, if the process were unable to escape the jail. The reality is that it can, so you should never treat a **chroot** as security measure. This doesn't mean **chroots** are useless: they come handy in packaging, for example. **Chroot**s are just not what they aren't – Linux alternatives to FreeBSD jails or Solaris Zones (look at namespaces(7) instead).

Command of the month: id

Now we know that user and group IDs come in three flavours. To obtain them in code, you use the system calls summarised in the boxout. But how do you know who you are, in a shell?

The **id** command is the answer. When run with no arguments, it prints a summary of your credentials:

```
$ id
uid=1000(val) gid=1000(val) groups=1000(val),4(adm),10(
wheel),14(uucp),90(network),91(video),92(audio),93(optic
al),98(power),108(vboxusers),150(wireshark),991(docker)
```

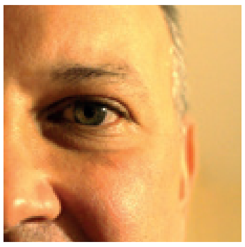
Quite often, you'll find a recommendation to supply the **-a** command line switch. In fact, this does nothing in Linux and is retained for compatibility only.

Alternatively, you can get raw identifiers (eg for use in shell scripts). **id -u** shows your effective user ID, **id -g** prints your effective group ID, and **id -G** lists supplementary groups. Add **-r** (eg **id -ru**) to print real identifiers instead. Note there's no way to get saved set-ID identifiers. You can also get IDs for another user in the system with **id <username>**.

Optionally, **id** can also print security context on SELinux-enabled kernels. Use **id -Z**.

/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!

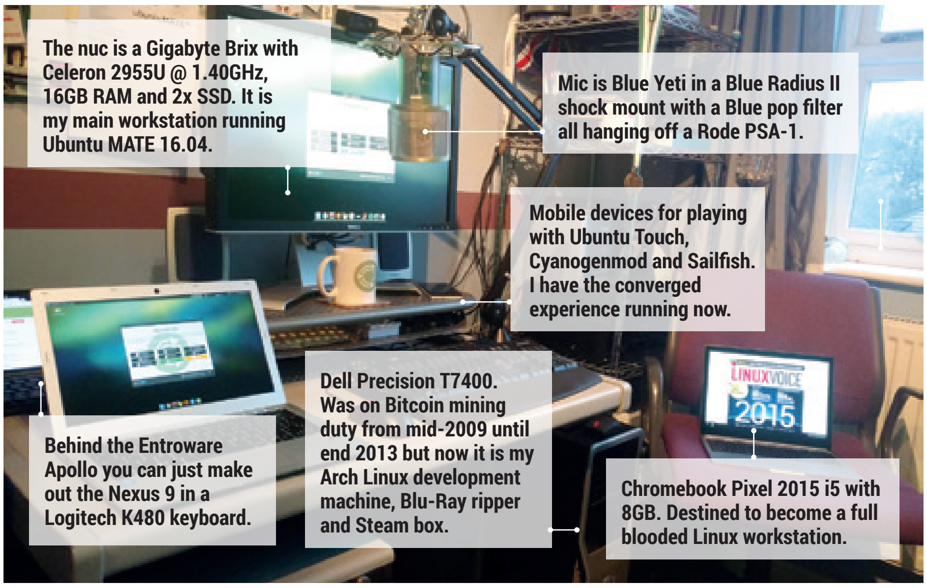
Cast your mind back to 1995, if you can. Robson and Jerome dominated the music charts, the Pokémon craze was in full swing and Nicholas Cage was seemingly in every film released – but it wasn't all grim. Linux was still rather obscure, but Peter Mattis and Spencer Kimball started working on something that would help change that – a software project known as *The Gimp*.

It seems remarkable to remember, but in five or six years it became a robust and remarkable tool that was actually superior to Adobe's *Photoshop* in terms of features and standards compliance. So what went wrong?

It is fair to say the project lost momentum. While the nascent Linux 'Enterprise' players were happy to fund developers to work on kernel features, filesystems and the like, graphics software was seen as, at best, non-essential. Hampered by a lack of experienced devs and hamstrung by technical debt (*Gimp* was, and still is, RGB only, in a world that wanted CMYK and more) it spent many years seeing only superficial updates.

The solution was a comprehensive retooling around a new graphics engine (*GEGL*) which can take better advantage of modern processors and address the fundamental restrictions of the original design – it's no less than a complete rewrite from the inside out.

Twenty years on, although it may seem like not much has changed recently, *Gimp* is poised once again to dazzle. And if you want to help it get there faster, you can always donate:
https://www.gimp.org/donating



The nuc is a Gigabyte Brix with Celeron 2955U @ 1.40GHz, 16GB RAM and 2x SSD. It is my main workstation running Ubuntu MATE 16.04.
Mic is Blue Yeti in a Blue Radius II shock mount with a Blue pop filter all hanging off a Rode PSA-1.
Mobile devices for playing with Ubuntu Touch, Cyanogenmod and Sailfish. I have the converged experience running now.
Behind the Entroware Apollo you can just make out the Nexus 9 in a Logitech K480 keyboard.
Dell Precision T7400. Was on Bitcoin mining duty from mid-2009 until end 2013 but now it is my Arch Linux development machine, Blu-Ray ripper and Steam box.
Chromebook Pixel 2015 i5 with 8GB. Destined to become a full blooded Linux workstation.

MY LINUX SETUP
MARTIN WIMPRESS

MATE hacker, Ubuntu MATE co-founder and Ubuntu Podcaster.

Q What version of Linux are you currently using?

A Ubuntu MATE Xenial Xerus development branch (what will become 16.04), on my Entroware Apollo and cheap Celeron nuc. Arch Linux on my Dell Precision T7400. Android on my Nexus 9 and moto X Style. ChromeOS on my Chromebook Pixel 2015 and Acer Chromebook C720. Sailfish 2.0 on my Jolla phone and a Nexus 4, Nexus 7 and bq Aquaris E5 all running Ubuntu Touch.

Q And what desktop are you using at the moment?

A MATE on all the proper Linux workstations. I really like LXQt as well.

Q What was the first Linux setup you ever used?

A Yggdrasil LGX in 1994. I was using Unix and Microsoft Xenix at work and really wanted a Unix for home but couldn't afford SCO UNIX. Once I found Linux I ditched OS/2 and haven't looked back. After Yggdrasil I used Slackware (1996–1998), Red Hat, CRUX, Fedora, Ubuntu (2004–2011), Arch Linux (2011 till present) and Ubuntu MATE since I started making it.

Q What Free Software/open source can't you live without?

A My favourite piece of open source software is OpenSSH. I'm not looking forward to Theresa May banning it but have been brushing up on my Telnet and FTP skills for when the cryptpocalypse happens.

Q What do other people love but you can't get on with?

A First-person shooters. Ever since I first sat in a hydraulic *Outrun* while on holiday in the summer of 1987 I've been hooked on racing games.

PIMORONI

http://pimoroni.com http://pimoroni.de

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).