FIREFOX: DISCOVER ITS HIDDEN FEATURES

LINUXVOICE

December 2015 FREE SOFTWARE | FREE SPEECH www.linuxvoice.com

› The best kids' software

› Build a DIY Oculus Rift

FREEDOM!

Switch to 100% Free Software today

- Purge your PC of proprietary software
- Get complete privacy and security
- Be free as Richard Stallman intended!

LIBREBOOT X200
Certified by the Free Software Foundation – could this be your next laptop?

ANDROID ANATOMY
Discover the incredible power in your pocket [ooh behave!]

32 PAGES OF TUTORIALS

INTERVIEW
ALLISON RANDAL
The president of the Open Source Initiative talks Perl, linguistics, and the Free Software/Open Source schism

DESKTOP
GNOME 3.18
You're going to love the shiny, slick desktop environment that beats Apple's OS X at its own shiny, slick game

FREE SOFTWARE | FREE SPEECH
ISSN 2054-3778
December 2015 £5.99 Printed in the UK

DATABASES › GITLAB › GARBAGE › ADA & MORE!

ANDREWS & ARNOLD LTD

will make you the

LINE KING

BE THE MASTER OF YOUR OWN TELEPHONE

SIP2SIM® from Andrews & Arnold allows you to treat your mobile handset as a SIP endpoint, freeing you from your desk and without the need of a smartphone app. Insert the SIM into your phone, point it to your Asterisk, FreeSwitch or other SIP server (or a commercial SIP service) and experience the reliability and call quality you're used to on a mobile, but with the flexibility of a SIP handset.

No minimum term. SIM £5+VAT and £2+VAT per month. Calls from 2p+VAT per minute.

Call 033 33 400 220, email sales@aa.net.uk or visit www.SIP2SIM.uk to find out more

HAKUNA MATATA

DANGER: GLOBAL WARNING

The December 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 doesn't make it any easier to accept, but it's no real surprise that Volkswagen has been caught fixing its emissions software. And it's probably just the tip of the iceberg. But if there is a positive aspect to this revelation, it's that Volkswagen has given us the perfect example of the dangers posed by some proprietary software and the unexpected consequences of overarching legislation like the DMCA in the United States or the EUCD in Europe.

But this is also an example that can be used to illustrate the many advantages of open source without any of the ideological baggage, and this is now more important than ever, as technology seeps into defibrillators, insulin pumps and autonomous cars. It's unrealistic to think that proprietary software will cease to exist, but it may give companies like Volkswagen another reason to try open source.

Graham Morrison
Editor, Linux Voice

What's hot in LV#021

ANDREW GREGORY

Building your own 3D head tracking system with a few spare parts and a Linux box sounds like something from the A-Team. But we've done it with just twelve easy steps and lots of images. **p80**

BEN EVERARD

The Electronic Freedom Foundation gets a lot of press and does some great work in the USA. Which is why it's great to be able to highlight a similar project based in the UK – the Open Rights Group. **p22**

MIKE SAUNDERS

The continuation of Juliet Kemp's wonderful Olde Code series (this month: Ada!) is a fascinating read for both old-school charm and for the modern languages that owe so much to the past. **p90**

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, Robin Gower, Juliet Kemp, Vincent Mealing, Simon Phipps, Les Pounder, Mayank Sharma, Valentine Sinitsyn.

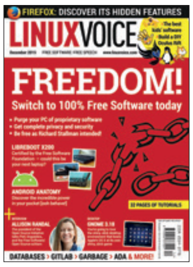
Linux Voice is different.
Linux Voice is special.
Here's why...

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

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

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

SUBSCRIBE ON PAGE 56



Contents

Software is born free, but everywhere it's in chains. Blame the cheesemakers.

Regulars

News 06
The coming and goings on planet Linux, including our assessment of the Volkswagen mass fraud, encapsulated for your reading pleasure.

Distrohopper 08
Manjaro, GNU Hurd, Calculate Linux and the BSD flavours hitting your hard drive (or virtualisation platform) right about now.

Speak your brains 10
Us Free Software types seem to be obsessed with privacy, with good reason. You're not alone in your mistrust of Google.

Subscribe! 12/56
Save money, get the magazine delivered to your door and get access to 21 issues of Linux Voice, in lovely DRM-free PDFs.

Secrets of Firefox 26
The flagship application in Free Software is facing competition from Chrome, but still has a few tricks up its furry sleeve.

Core Tech 94
Programmers these days don't have to think too much about memory management, but it never hurts to know the basics. So we give you: garbage collection.

Geek Desktop 98
Mr Libreboot, Francis Rowe, doesn't like KDE. Mr Bushmills, Nick Veitch, doesn't like humans. Like ebony and ivory, they live together in perfect harmony on page 98.

Cover Feature

14

FREEDOM!

Switch to 100% Free Software today

For privacy, for security, and to make Richard Stallman happy – you too can ditch proprietary software completely and go 100% free today.

Interview

34

Allison Randal

The boss of the OSI talks Open Source and the schism at the heart of Free Software.

Feature

22

Inside ORG

Too busy to write to your elected representative? Don't worry, the Open Rights Group will do it for you!

FAQ

OpenStack 32
Take the modular, scalable principles of Unix and apply them to the cloud. Unlimited power!

Group Test

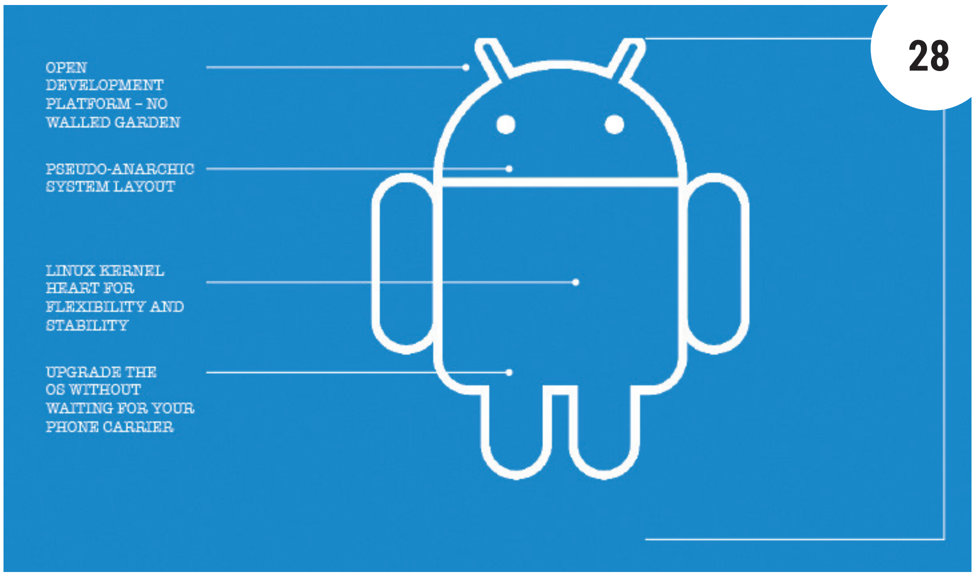
Educational software 50
Find the best games, projects and whimsy to keep kids entertained while stimulating their brains.

FOSSPicks

... yes, Fosspicks! 58
These six pages contain more freedom than the Declaration of Arbroath. That's lot of freedom.

SUBSCRIBE ON PAGE 56

Feature


28
OPEN DEVELOPMENT PLATFORM – NO WALLED GARDEN
PSEUDO-ANARCHIC SYSTEM LAYOUT
LINUX KERNEL HEART FOR FLEXIBILITY AND STABILITY
UPGRADE THE OS WITHOUT WAITING FOR YOUR PHONE CARRIER

Android anatomy
Install a custom OS on your Android phone and cut out all the rubbish you don't want.

Reviews

Bitwig Studio 1.2
At last, there's some competition on the desktop audio production scene, in the shape of Bitwig Studio. But how does its latest Linux version hold up?

Gnome 3.18 43
The Gnome Foundation scores a huge 6 with its documentation, polish and panda emojis.

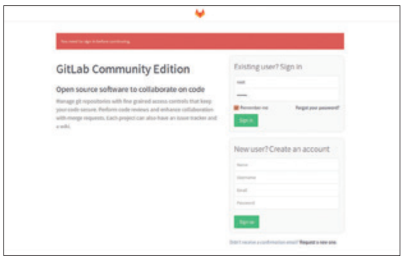
Unity 44
Create games for any platform you like (as long as you don't mind a tangled web of licences).

Devolo 1200+ 45
Extend the range of your home Wi-Fi throughout your sprawling Dordogne château.

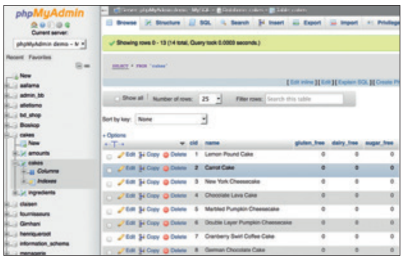
Gaming on Linux 46
This month, a spiritual successor to the awesome *Syndicate*, one of the coolest games ever to grace the Amiga (spiritual predecessor to Linux).

Books 48
We're going back to basics without book selection this month, with a look at electronics, programming and the noble art of darts.

Tutorials


GitLab Community Edition
Open source software to collaborate on code
Existing user? Sign in
New user? Create an account

GitLab 66
Don't depend on Sourceforge – host your world-beating project yourself!



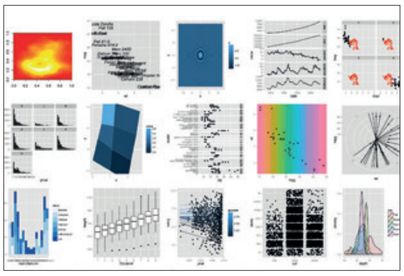
MariaDB 68
In the kingdom of the database, the man who understands MariaDB is king...

Raspberry Pi 72
Link a funky HAT with some Python and you're got yourselves a newsreader.

OpenVAS 76
Scan your network for vulnerabilities automatically, saving time and money.

Hardware hacking 80
Craft a 3D motion tracking headset out of eBay's cheap cast-offs.

Coding



GNU R 84
Use the power of maths to reveal how safe your favourite Free Software project is.

Rust 88
Launch yourself into the next web with Mozilla's new programming language.

Olde Code: Ada 90
The US military has given us many things; our favourite might just be this language.

NEWSANALYSIS

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

Opinion

Code, law and digital restrictions

The VW scandal reveals the self-defeating nature of the laws we use to protect ourselves

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.

The revelation that Volkswagen and other vehicle manufacturers had apparently been cheating the vehicle emissions inspectorates of the world has brought back into focus three observations from the free culture movement. The lesson they teach? We must rethink the criminalisation of citizen curiosity.

Code Is Law

Law professor and US presidential candidate Lawrence Lessig first pointed out the impact of technology on the administration of law in his book *Code and Other Laws of Cyberspace* in 1999, where he coined the aphorism "Code Is Law". By this he meant that the real regulation of the world arises not in the letter of the legislation, but rather in the way programmers interpret it and then express their interpretation in the software that applies the law. More casually, the software in the systems around us is what creates our context and environment.

Trust – but verify

If code is law, how can society remain transparently governed? Law professor and former developer Eben Moglen – one of the key minds behind the GPLv3 and founder of the Software Freedom Law Centre – has frequently pointed out that proprietary software is inherently challenging to the trust society has in its administrators and suppliers. When an agency or a supplier refuses to allow scrutiny of its source code, only its own staff can check it for errors. Open source is vital for the software used to regulate the public good, because without it, we are forced to trust blindly those who are theoretically serving us. When it comes to critical infrastructure, this is especially challenging; as Moglen has said, "Proprietary software is an unsafe building material. You can't inspect it."

Digital restrictions

If code is law, and if our trust is only well placed if we can verify, then the observations of science fiction author and digital rights campaigner Cory Doctorow illuminate the third facet of the situation. Technologically illiterate legislators in thrall to the entertainment industry have mistakenly passed laws that criminalise the circumvention of "technical measures" that obfuscate software. As any geek knows, measures that artificially obscure technical details can always be worked around, often trivially easily. Business models that depend on such measures are thus fatally flawed – unless shored up by draconian laws.

That's what the DMCA in the USA and laws implementing the EUCD in Europe have done. They make "circumventing" a "technical measure" a crime, mostly regardless of context. Sadly these laws omit exceptions for legitimate activities, with the result that activities the legislators never imagined have fallen into the dragnet of anti-circumvention laws. Legislators thought they were protecting the music and movie industries against "pirates"; instead, they have criminalised the scrutiny of emissions control software in vehicles, protected the wasteful monopolies of printer ink manufacturers and created an avenue for unseen manipulation of voting machines. As Doctorow has said, "DRM changes your computer from 'What can I do for you, sir' to 'I can't let you do that, Dave.'"

Vehicle manufacturers gamed emissions testing because they could. Undoubtedly their games should be penalised. But we would be foolish if that were the only fix we furnished. The gaming of vehicle emissions hid behind the DMCA and the European Copyright Directive, and is just one instance of unethical practice enabled by misguided laws that try to criminalise an arbitrary kind of programming. The right response to VW is to rethink digital restrictions and make them reflect the real contours of the digital landscape – not the imaginary ones the media industry dream could exist.

> The right response to VW is to rethink digital restrictions and make them reflect the real contours of the digital landscape

> As any geek knows, measures that artificially obscure technical details can always be worked around, often trivially easily

LibreOffice • Gnome • Raspberry Pi • CHIP • Microsoft • Wikimedia

CATCHUP

Summarised: the biggest news stories from the last month

1 Italy switches 150,000 PCs to LibreOffice

While we'd like to see more governments and companies migrate to Linux, any major moves towards Free Software (even on Windows) deserve applause. Italy's Ministry of Defence has announced that it will transition 150,000 PCs to *LibreOffice*, following a law passed in 2012 which states that free and open source software should be the default option in the country's public administration bodies. So perhaps a switch to Linux will come...
http://tinyurl.com/nhsmzmf

2 Gnome 3.18 released with new goodies galore

After six months of hacking, including 25,112 source code changes from 772 contributors, version 3.18 of the Gnome desktop is ready for the masses. It includes Google Drive integration, firmware updates through the Software Tool, automatic screen brightness changing (depending on light levels around you), touchpad gestures and new Calendar and Characters applications. Expect it in the next round of distro releases.
www.gnome.org

3 Pi touch display is here

Yes, the Raspberry Pi now has its very own official touch display. It's a 7-inch 800x480 screen with capacitive touch capabilities, and can connect to the Pi via HDMI, DPI, DSI and DBI interfaces. The cost: £48/$60.
http://tinyurl.com/pidisplay

4 $9 CHIP single-board computer starts shipping

The Raspberry Pi is about to get a competitor, and a crazily cheap one at that. CHIP – the "world's first nine dollar computer" – was successfully Kickstarted earlier in the year, and units are now being sent out to early backers. The CHIP has a 1GHz ARM CPU, 512MB of RAM and 4GB of onboard storage, and of course it runs Linux. It's considerably weaker than the Pi 2, but still capable of many jobs – we'll get one in for review soon.
http://tinyurl.com/q54s9kd

5 Raspbian updated to Debian 8, aka "Jessie"

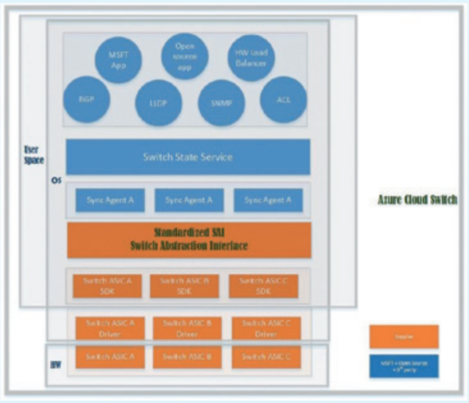
Raspberry Pi owners rejoice: a new version of Raspbian, the popular Debian-based distro, is here. This time it's based on Debian 8 ("Jessie"), and brings a boatload of improvements. Most notably, it boots to the desktop by default now, and includes a GUI setup tool as an alternative to *raspi-config*. This new Raspbian release features plenty of optimisations too, to squeeze more performance out of the relatively low-spec machines. Download here:
www.raspberrypi.org/downloads

6 LibreOffice celebrates its fifth birthday

More good news for the flagship open source office suite. On 28 September 2010, a small group of hackers and FOSS fans decided to split off from *OpenOffice.org* and create a new project with a more rapid pace of development. *LibreOffice* has come a long way in that half-decade: it has replaced *OpenOffice.org* (now *Apache OpenOffice*) in virtually every major distro, and is getting faster and more polished with every release.
https://t.co/AvQBPbtkx2

7 Microsoft makes an OS based on Linux

What a time to be alive: Microsoft has unveiled Azure Cloud Switch, a "cross-platform modular operating system for data centre networking built on Linux". So it's a very specialised stack and a long way from a desktop distro, but underscores the changing attitudes within Microsoft towards Linux and Free Software. We're still sceptical, but if Microsoft contributes useful code back to FOSS projects, we'll give them some kudos.
http://tinyurl.com/qamw5se



8 Wikimedia starts new mapping project

The Wikimedia Foundation hosts many big-name sites like Wikipedia, Wikibooks and Wikivoyage, and is now collaborating with the rather awesome OpenStreetMap team to create a new map rendering system. The Foundation's goal is to "encourage our community to create tools and integrations to develop new ways for readers and editors to discover Wikimedia content", and you can see the early development work here:
www.mediawiki.org/wiki/Maps

DISTROHOPPER

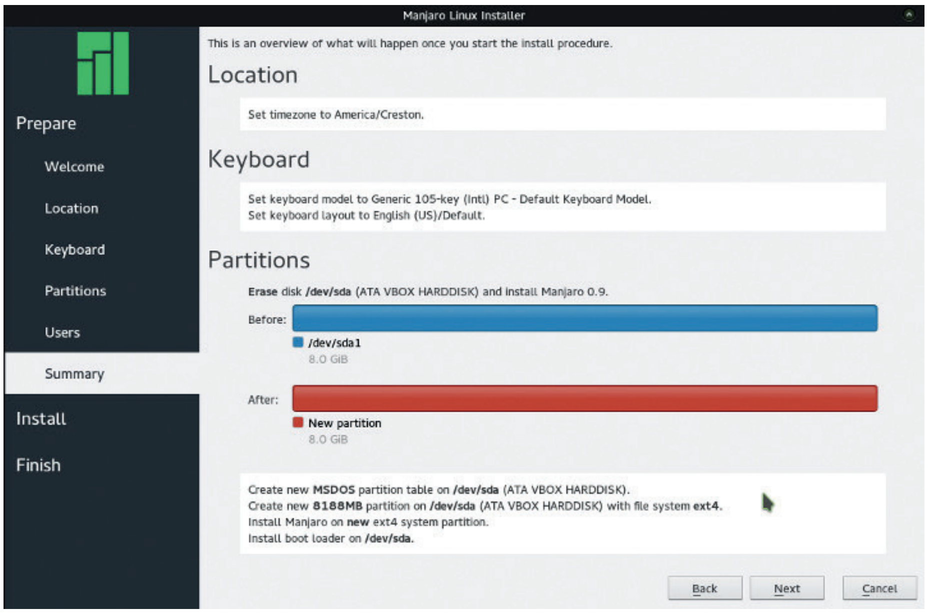
What's hot and happening in the world of Linux distros (and BSD!).

Manjaro 15.09

User-friendly Arch goodness.

We use a bunch of different distros at Linux Voice Towers, and one of our favourites is Arch. But setting it up manually can be a chore sometimes, especially when we get a new bit of kit and want to put Linux on it as quickly as possible, so Manjaro has become one of our favourites. It takes all the good bits of Arch – the speed, the customisability – and adds a click-and-go installer, an attractive desktop setup and other polish. It still lets you tinker with the system and provides access to the hugely useful Arch User Repository – but it doesn't take as long to set up and provides extra codecs and drivers out of the box.

The previous release of Manjaro was numbered 0.8.13, but now the team has switched to date-based versioning – hence 15.09. And the biggest change here is the inclusion of *Calamares*, a graphical installation tool. Beforehand, Manjaro's installer was a text-based affair that did an adequate job, but wasn't really all that user-friendly. *Calamares* is much more polished and user-friendly, and while it's still undergoing development, the Manjaro team has deemed it "stable enough for use on production systems".


Manjaro Linux Installer

Prepare
Welcome
Location
Keyboard
Partitions
Users
Summary
Install
Finish

This is an overview of what will happen once you start the install procedure.

Location

Set timezone to America/Creston.

Keyboard

Set keyboard model to Generic 105-key (Intl) PC - Default Keyboard Model.
Set keyboard layout to English (US)/Default.

Partitions

Erase disk **/dev/sda** (ATA VBOX HARDDISK) and install Manjaro 0.9.

Before: /dev/sda1 8.0 GiB

After: New partition 8.0 GiB

Create new **MSDOS** partition table on **/dev/sda** (ATA VBOX HARDDISK).
Create new **8188MB** partition on **/dev/sda** (ATA VBOX HARDDISK) with file system **ext4**.
Install Manjaro on **new** ext4 system partition.
Install boot loader on **/dev/sda**.

Back Next Cancel

Thanks to *Calamares*, Manjaro now has a graphical installer – although text-mode is still available.

If you've heard us waxing lyrical about Arch but never had the time to install it, give Manjaro a go. It's an excellent way to get the benefits of Arch without the learning curve.

Calculate Linux 15

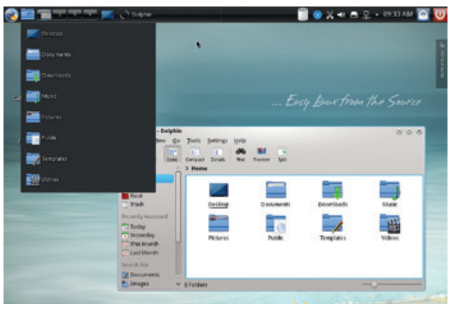
Slick Gentoo-based distro for desktops, servers and media.

Distros based on Debian, Ubuntu and Fedora are ten a penny these days, but one distro that has relatively few derivatives is Gentoo, the source-based distro that was once hugely popular among tinkerers and power users.

Gentoo is still a good distro, but it requires some effort to get working, so Calculate Linux aims to make the job easier. It's 100% compatible with Gentoo and features the same rolling-release model – so users always get the latest apps without having to perform major upgrades every six or 12 months – but with ready-built KDE, Xfce and Mate desktops plus configuration tools and other goodies.

Calculate Linux 15 is a major update providing improved reliability and performance during package updates, while the Calculate Utilities suite has been updated with new system build tools, and a GUI has been added to Calculate Console.

It's a well-presented distro with plenty of documentation, and available as a live DVD image that can be installed to hard drives and USB flash keys. We have some concerns about the long-term stability of rolling-release distros, when a system package update can potentially break the system or introduce new problems but, as Arch shows, it's possible to get it right most of the time. To grab the latest release, pop over to the project's website at **www.calculate-linux.org**.



Calculate is available with KDE, Xfce and Mate, along with a Directory Server for businesses.

News from the *BSD camps

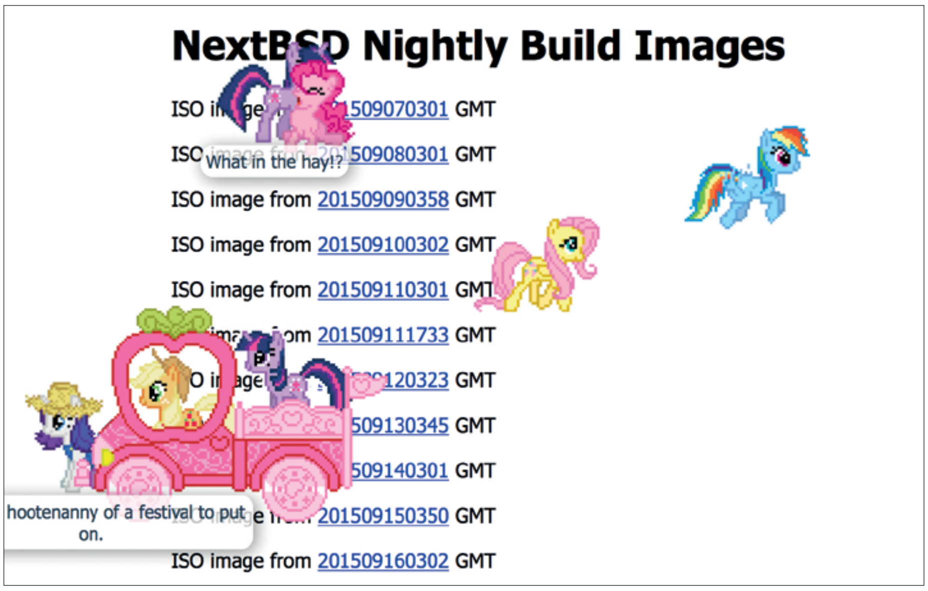
What's going on in the world of FreeBSD, NetBSD and OpenBSD.

FreeBSD is well known for being a rather conservative project, waiting for new features and technologies to mature before rolling them in to the main source code tree. This pace of development has served FreeBSD well over the years, but some developers are looking to make more ambitious changes to the operating system. NextBSD (**www.nextbsd.org**) is a "science project" that aims to "adapt some of the more interesting Open Source technologies from Darwin/OS X to FreeBSD".

So it's not a fork of FreeBSD, but rather a distribution that adds certain bleeding-edge features onto the main codebase.The most notable of these features is *launchd*, Apple's init system and service management framework, which is responsible for starting and stopping scripts, background daemons and applications. Despite being part of OS X, *launchd* is open source, like various low-level components of that operating system.

But why does FreeBSD need a new service manager? Hasn't the traditional BSD-style init worked well over the years? Well, yes, but *launchd* aims to make booting faster by starting services in parallel, and provides much more flexibility so that services can be started when mount points appear, sockets are opened, or filesystem changes are made.

Some developers are looking to the future and saying that FreeBSD needs to adapt to stay competitive and relevant – so if there's useful open source code in Mac OS X that's worth nabbing, why not?


NextB[illegible]D Nightly Build Images

ISO i[illegible]ge[illegible]1509070301 GMT
ISO [illegible]1509080301 GMT
ISO image from 201509090358 GMT
ISO image from 201509100302 GMT
ISO image from 201509110301 GMT
[illegible]ima[illegible]om 201509111733 GMT
[illegible]O i[illegible]age[illegible]120323 GMT
[illegible]509130345 GMT
[illegible]509140301 GMT
[illegible]e [illegible] 201509150350 GMT
ISO image from 201509160302 GMT

NextBSD ISOs are available from **http://build.nextbsd.org/data** – and you get free ponies as well.

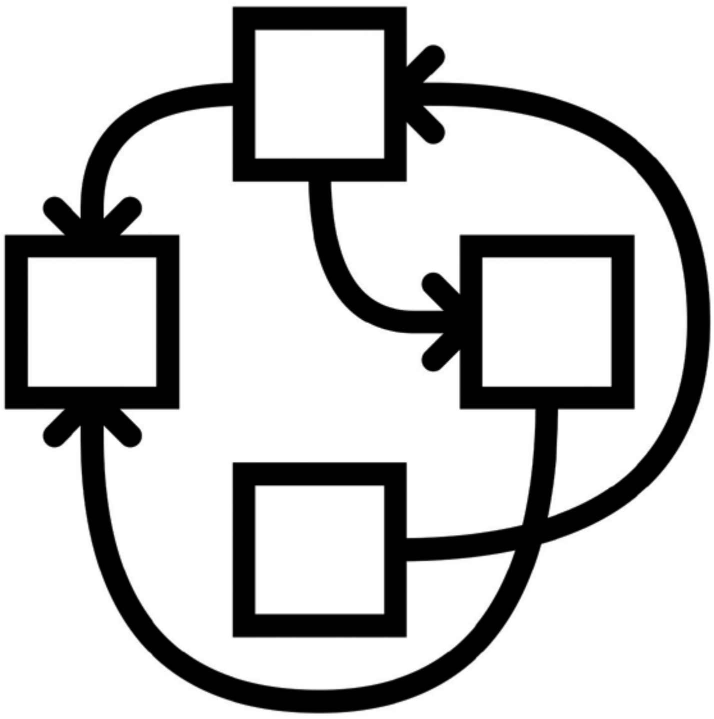
What's the deal with: GNU Hurd?

GNU/Linux, the GNU operating system with the Linux kernel, is what we use today. But the GNU project started in 1984, seven years before Linus Torvalds announced his first kernel release, and for many years it was expected that GNU would have its own kernel. But this component of the operating system wasn't a priority for the GNU developers; they were focusing on a C compiler, text editor and system libraries – things they could run on other Unix platforms before theirs was fully ready.

However, in 1990 the GNU team decided they really needed to get cracking on a kernel, and Hurd was born. The name is a doubly-recursive acronym: "Hurd" stands for "Hird of Unix-Replacing Daemons", while "Hird" stands for "Hurd of Interfaces Representing Depth". Try explaining that to someone at the pub…

But anyway. Hurd's most notable feature is that it's a microkernel, where only a small portion of the code (memory management, process scheduling and some hardware drivers) runs close to the hardware. The rest of the kernel runs in userspace, like a normal program, so if a network stack, filesystem driver or other feature suffers a crash, it shouldn't take down the rest of the kernel.

Of course, the need to regularly switch between kernel and user modes impacts performance, and GNU Hurd has lagged way behind Linux in development activity. Even Richard Stallman has said that Hurd is not a priority for the GNU project nowadays, because the Linux kernel does a good job. Still, there are GNU distributions using the Hurd kernel, such as Debian GNU/Hurd (**www.debian.org/ports/hurd**). For more information, see **www.gnu.org/software/hurd**.


Hurd's design makes it potentially more robust than Linux, but it has far fewer active developers.

YOUR LETTERS

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

STAR LETTER

THINK BEFORE YOU SEARCH

DuckDuckGo's bangs are great, and it's easy to see why people love them. But they underline a bad habit we as users have fallen into. Back in 2012 Eben Moglen [founder of the Software Freedom Law Centre] gave a talk at Republica, where he used the term "Superfriend" – that third party in the middle who knows everything because we willingly inform them in exchange for convenience. By depending on bangs we are placing DuckDuckGo in exactly that position.

"!g example" talks to DuckDuckGo and Google. "!w example" talks to DuckDuckGo and Wikipedia. "!r example" talks to DuckDuckGo and Reddit. This isn't malicious, it's just how it works – DuckDuckGo needs to know what you're looking for so it can interpret the bang.

I'm not advocating people don't use DuckDuckGo – you should use it. But bangs aren't new – *Firefox*, *Chrome*, *Konqueror*, and pretty much every browser implements some form of them. For *Firefox* and *Chrome* they are "Keywords", for KDE browsers they are "Web Shortcuts" – and in all of them you can add as many as you like and set the prefix to anything you like.

The solution to the private search problem isn't to use a private search engine – to create a second, more responsible Google – because a single point of failure only takes a single gagging order. The solution is to change how we search – to search smarter. To give as little information as possible to search engines, and only ever what is necessary. And it's going to be a hard, long, and difficult fight to make that change.

Patrick McDonough

Andrew says: Hmm, that's an angle I'd never thought of until now – use the minimum of keywords and click through the links until you find what you're looking for, rather than give the search engine all the keywords (I guess this is what you mean by searching smarter, but I'm not very smart). It's easier said than done though, and really only works if you know in advance what you're looking for. With DuckDuckGo, at least we know that if the injunction comes and Google has to spill its guts to the police, there are alternatives around that we can continue to use. Hopefully one day we'll be able to show the political culture of mass surveillance to be what it is: paranoid, illiberal, useless and undemocratic.

ANDROID PRIVACY

Nothing stated in your article (issue 18, September 2015) allays my privacy concerns about using Android. I had a weather app on my tablet and a recent update wanted access to my identity. Why does a weather app need access to my identity? I've been using the app for about a year or more, so why does it now need to know who I am? I decided just to completely remove the app from my tablet.

I also realise that there are ways to combat app permissions, but such methods either entail not using the app or rooting your device which may void the warranty or cause issues with your mobile provider.

The only good thing I can say about Android vs iOS is that at least you know what permissions apps have. I so much want to have a smartphone as they are very useful devices, but I value my privacy much more. I'm very much hoping that an Ubuntu- or Sailfish-based device becomes available soon via my mobile provider.

Kurt Meyer

Andrew says: "Why does a weather app need access to my identity?" is a very good question indeed. I imagine if you were to ask the developer, the answer would be some guff about providing you with an enhanced user experience, seamlessly integrating with your digital lifestyle and online experience. Which seems a bit much when all you want to know is whether to take an umbrella to work. I also feel uncomfortable giving too much information away, and now that my Nexus 5 is about to pass like tears in rain, I really want to go back to the simpler days when Snake didn't need to know your mother's maiden name and blood type.

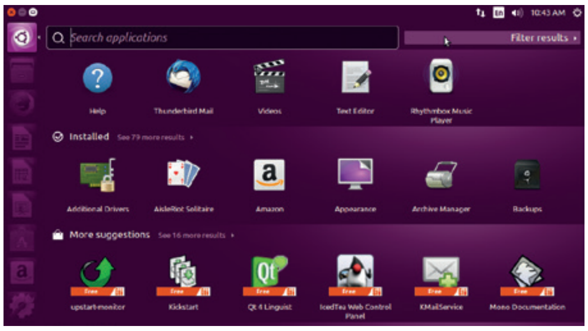
CINNAMON

I have three old Dell 170L 32-bit machines with Celeron processors. One has Ubuntu 12.04 on it, another Kubuntu 12.04; in both cases 14.04 will barely run, with some parts of Kubuntu not working at all on these machines, yet the third machine runs happily on 13.1 Mint Cinnamon. This seems to be an exception to your comment 'Avoid if you are running old hardware'.

On a different issue; after reading your September issue I tried Mageia 5 (this time on a Dell 210L machine) . Despite the impressive looks I found it did not work completely 'out of the box' :

Problem 1 – It would not download updates; this was traced to two pre-ticked boxes telling the system to look in the optical drive for updates, where it stopped looking further. Unticking these boxes solved the problem .

Problem 2 – When trying to use the terminal with a sudo command I received a message telling me that I was not on the sudo'ers list and had been reported. The reason for this is because the sudo'ers list did not exist. The system installation did not include sudo (or Grub) software, this has to be downloaded and installed after installation of the main system .



Ubuntu isn't just a flashy desktop – it's hours of unseen graft, ironing out hardware problems and making sensible choices. It's great.

The point being that for people like me who just want a system that works without problems, any new system must be complete enough for all the basic functions the novice user is likely to perform to work, something that Ubuntu and Mint do very well. Once sorted, Mageia 5 appears to be a good system, although I can't help wondering what else they have missed .

John Bourne

Graham says: Working out of the box is a feature. It can't be overstated how important this is. Thanks for your observations John, and I'm glad your old machines are getting a new lease of life.

TUX: SHOULD HE STAY OR SHOULD HE GO? PTII

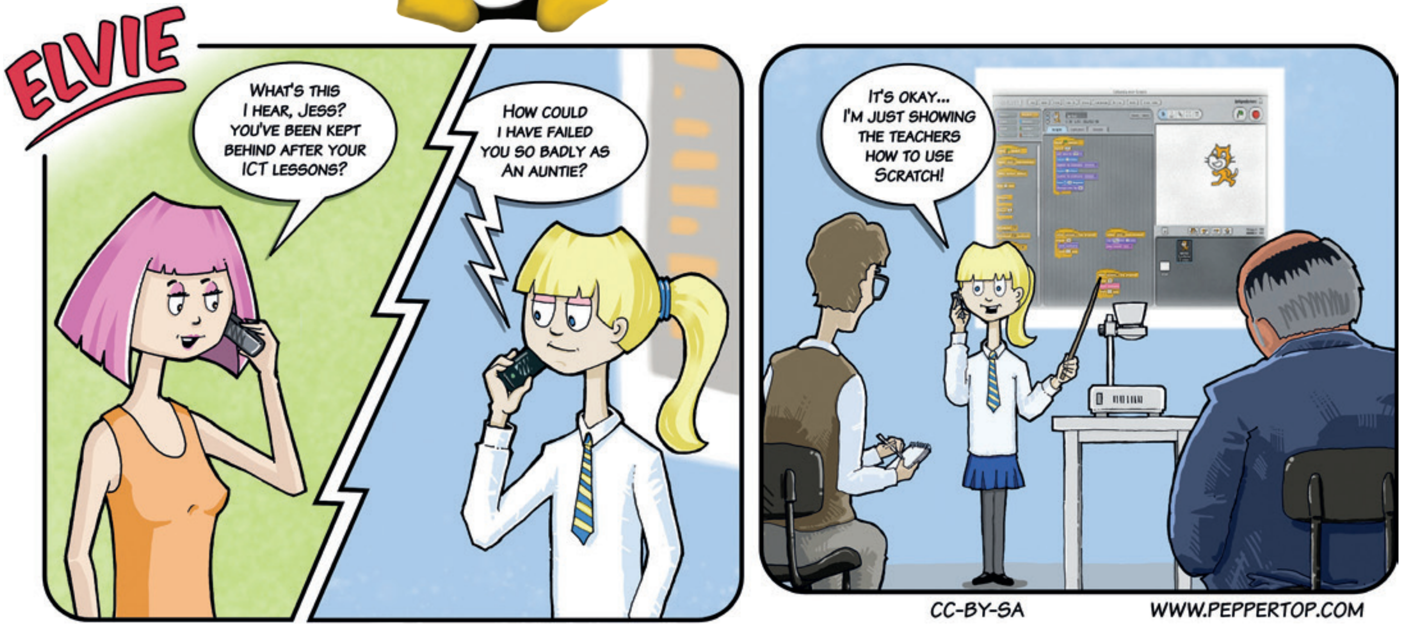
Back in the day, in 1995 when I first started using Linux, Tux fulfilled two very important functions that are still appropriate today.

Firstly, a unifying theme. I used Red Hat, Slackware, SUSE, and Debian – all variously branded, all Linux, and Tux was the link. Secondly, while we may have serious industrial-strength software to use, it is easy to take ourselves too seriously, so Tux was always meant to be a bit ironic in the 'GNU's not Unix' sort of way. Let's regain that idea of being able to laugh at ourselves again. Today, for me, he is a symbol of continuity – from the near 30-year-old me hacking Linux onto a 386dx66 with 4MB (yes children, 4 megabytes!) of RAM, to today and whacking a DVD into whatever and it just works.

Pass me my zimmer frame and false teeth.

Richard Rowe

Andrew says: The penguin is awful. We'll just have to agree to disagree on this one.


ELVIE
WHAT'S THIS I HEAR, JESS? YOU'VE BEEN KEPT BEHIND AFTER YOUR ICT LESSONS?
HOW COULD I HAVE FAILED YOU SO BADLY AS AN AUNTIE?
IT'S OKAY... I'M JUST SHOWING THE TEACHERS HOW TO USE SCRATCH!
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

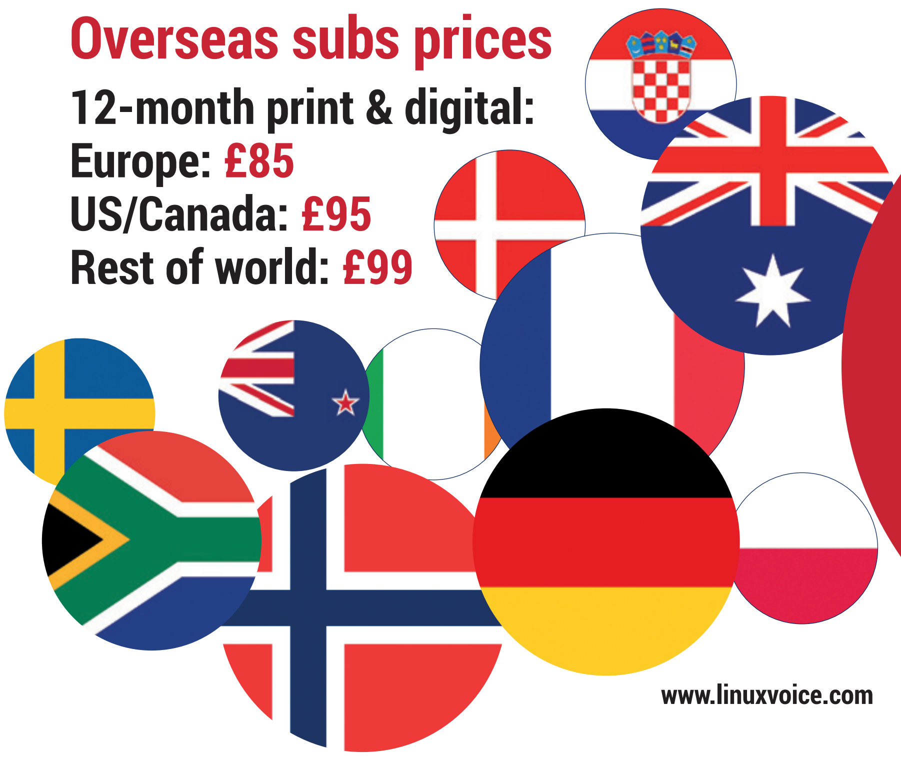
Save money on the shop price and get each issue delivered to your door

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

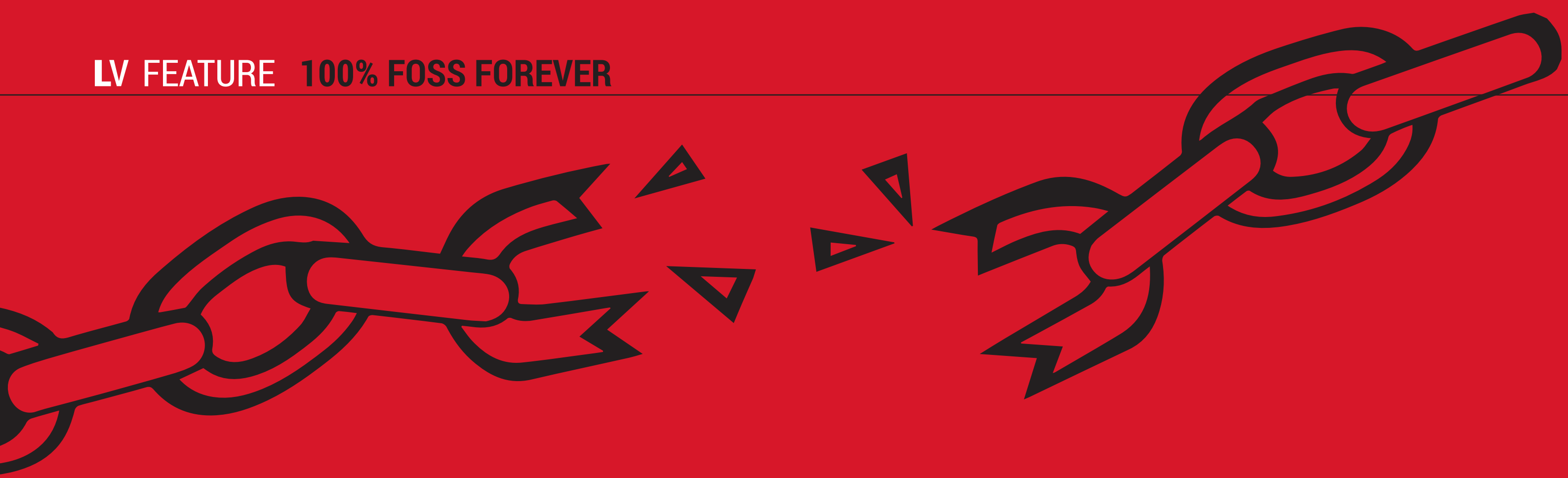
12-month print & digital:
Europe: £85
US/Canada: £95
Rest of world: £99



DIGITAL SUBSCRIPTION*

ONLY £38

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

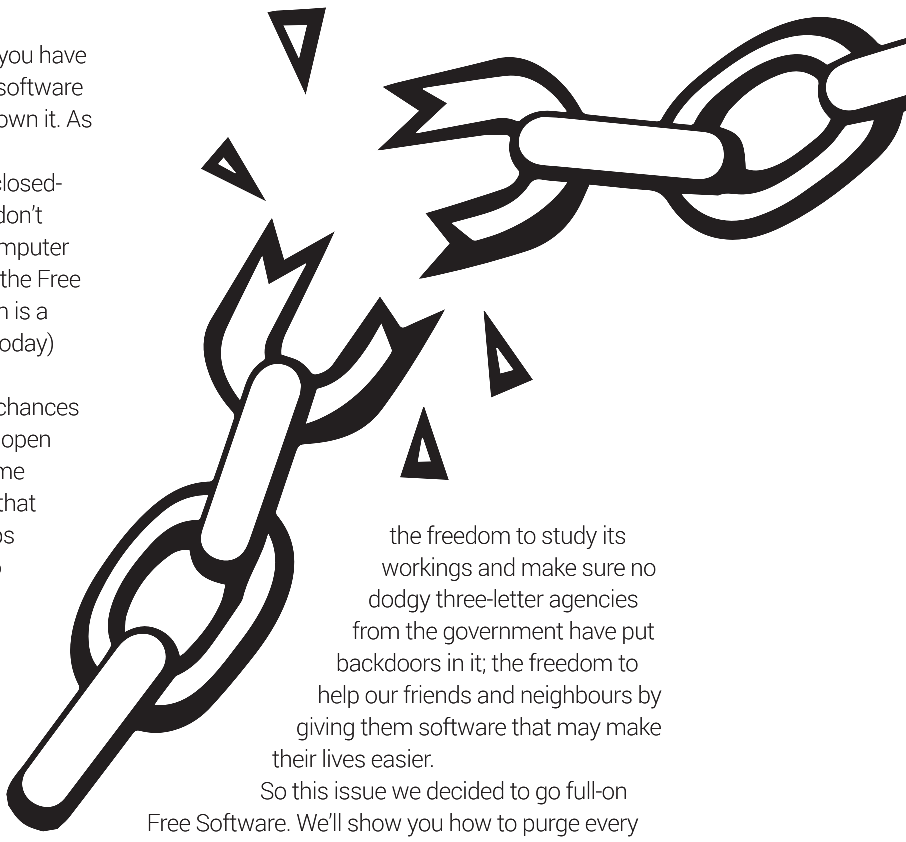


FREEDOM

Moving to 100% Free Software is essential in the fight for privacy, security and freedom. **Mike Saunders** explains all.

Who really owns your computer? If you have even the tiniest piece of non-free software installed, it's fair to say: you don't own it. As long as you're using software that you can't investigate, take apart and modify – that is, closed-source and proprietary software – then you don't really control your computer. Indeed, your computer controls you. Richard Stallman, who created the Free Software Foundation and GNU project (which is a huge part of the GNU/Linux system we use today) has been saying this for years.

Now, as you're reading this magazine, the chances are that you're already using mostly free and open source software – but there are probably some programs or online services that you rely on that are closed source and proprietary. These apps and services may seem harmless or claim to protect your data and privacy, but in a world where governments and corporations are performing dragnet operations to suck up all our data and correspondence, the only way to be truly secure is to run a 100% free software stack from the ground up.



It ain't about the money

If you're new to Linux, it's important to note what "free" means when we talk about Free Software. It's free as in speech – not as in beer. We GNU/Linux and FOSS fans don't particularly care about the zero price aspect (although it's good that we can save money for other things – such as beer). No, we care about the freedom it provides: the freedom to use, modify and share the software; the freedom to study its workings and make sure no dodgy three-letter agencies from the government have put backdoors in it; the freedom to help our friends and neighbours by giving them software that may make their lives easier.

So this issue we decided to go full-on Free Software. We'll show you how to purge every last proprietary byte from your machine, explore some distros that only include 100% FOSS, look at alternatives to proprietary online services, and take the awesome Libreboot laptop – which is also 100% FOSS right down to the BIOS, and Richard Stallman-approved – for a test run. When you've experienced total computing freedom, you (and your PC) will never want to switch back.

FIRST STEPS

What's free, what's non-free, and how you can purge your system of the latter.

Most software in a typical Linux distro is provided under a Free Software licence, but certain parts – such as hardware drivers and games – may not fall under the definition of Free Software. But what exactly is that definition? Richard Stallman is the best authority on this matter, we feel, and he says that a program can be classed as Free Software if it provides users with four "essential freedoms":

1 The freedom to run the program as you wish, for any purpose.
2 The freedom to study how the program works, and change it so it does your computing as you wish.
3 The freedom to redistribute copies so you can help your neighbour.
4 The freedom to distribute copies of your modified versions to others.

What's absolutely essential to fulfilling the above freedoms is access to the source code. Not just so you can look at it – also that you can change it and redistribute your changes. Now, the most successful and prominent Free Software licence is the GNU General Public Licence (GPL). This is a long and intricate document that guarantees the above freedoms and expands upon the fourth. The GPL states that derivative software – ie programs based on code from another GPL program – must retain the same licence and provisions for freedom. In this way the GPL is sometimes regarded as a "viral" licence, although we're not complaining when it spreads software freedom.

If you wanted to run a purely GPL GNU/Linux installation, however, you're out of luck. Many other major components, such as the X Window System and OpenSSH, have different licences, such as MIT and BSD. These also provide the four essential freedoms, but they don't force them on others. Some would argue that these licences are actually more free than the GPL, in that they have fewer restrictions; others say they don't protect the user's freedom enough. It's an argument that will no doubt go on for years.

> While most of the Linux kernel is released under the GPL, it also includes closed source firmware and drivers

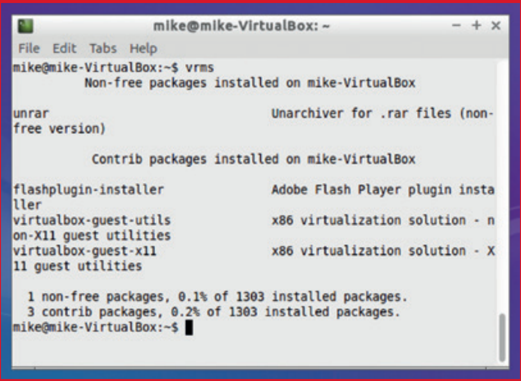
The Linux kernel is a special case: while the vast majority of its source code is licensed under the GPL and is therefore Free Software, it also includes various "binary blobs" – that is, closed source firmware and drivers for certain chips and hardware. Many users don't object to these and use them for pragmatic reasons, but as we mentioned at the start, they put the control of your computer in someone else's hands.

So a project called Linux-libre (**www.fsfla.org/ikiwiki/selibre/linux-libre**) has been developed by the Latin America branch of the Free Software Foundation. This removes all of the binary blobs from the Linux source code tree, and publishes the results as a separate branch. Consequently, 100% Free Software GNU/Linux distros can use this kernel without compromising any of the four essential freedoms.

Get Stallman to check your PC

Wouldn't it be great if Richard Stallman could come round to your house and personally inspect your PC for non-free programs? (Tip: have two cans of non-diet Pepsi ready, and if you have a parrot, even better – **http://tinyurl.com/rmsrider**.) Unfortunately Stallman is too busy to make sure we're not being led astray by proprietary software, but there's a program you can install that will scan your machine for programs that would furrow the great man's brow.

Vrms (Virtual Richard M Stallman, **http://vrms.alioth.debian.org**) scans Debian and Ubuntu-based installations for non-free packages, which is mightily helpful if you've just done a default installation and want to purge everything that isn't full Free Software. *Vrms* lists the non-free packages on your system, gives you explanations about licence issues if necessary, and then summarises the proportion of free to non-free software that you have installed. On a typical desktop system, chances are that you'll see the Adobe Flash Player plugin listed; if you want to remove this but still really need Flash support, try *Lightspark* (**http://lightspark.github.io**), an open source plugin. It can't handle all Flash content perfectly yet, but it's making good progress.



```
mike@mike-VirtualBox:~$ vrms
          Non-free packages installed on mike-VirtualBox

unrar                                   Unarchiver for .rar files (non-
free version)

           Contrib packages installed on mike-VirtualBox

flashplugin-installer                   Adobe Flash Player plugin insta
ller
virtualbox-guest-utils                  x86 virtualization solution - n
on-X11 guest utilities
virtualbox-guest-x11                    x86 virtualization solution - X
11 guest utilities

  1 non-free packages, 0.1% of 1303 installed packages.
  3 contrib packages, 0.2% of 1303 installed packages.
mike@mike-VirtualBox:~$ 
```

Use *Vrms* to get an at-a-glance overview of the non-free packages on your system.

Even when he's rafting down rivers, Richard Stallman is fighting to preserve the four essential software freedoms.

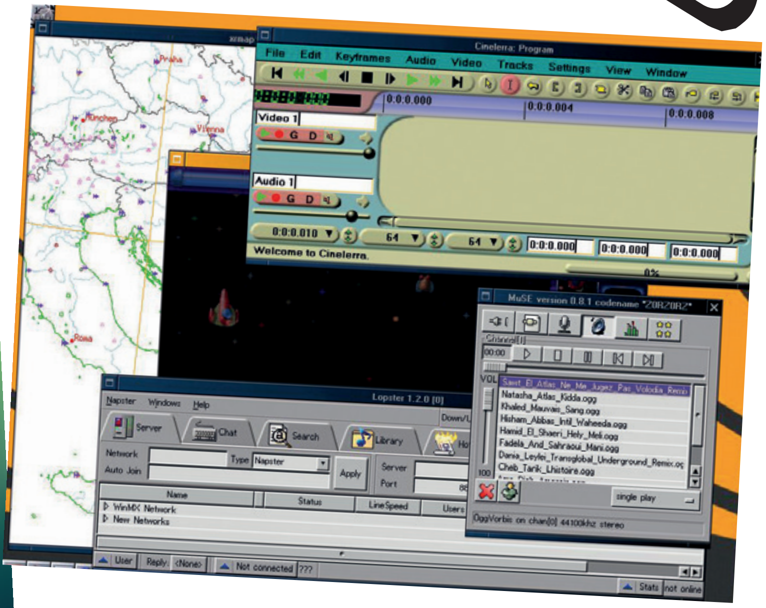
(CC-ND, from www.stallman.org/photos)

FULLY FREE DISTROS

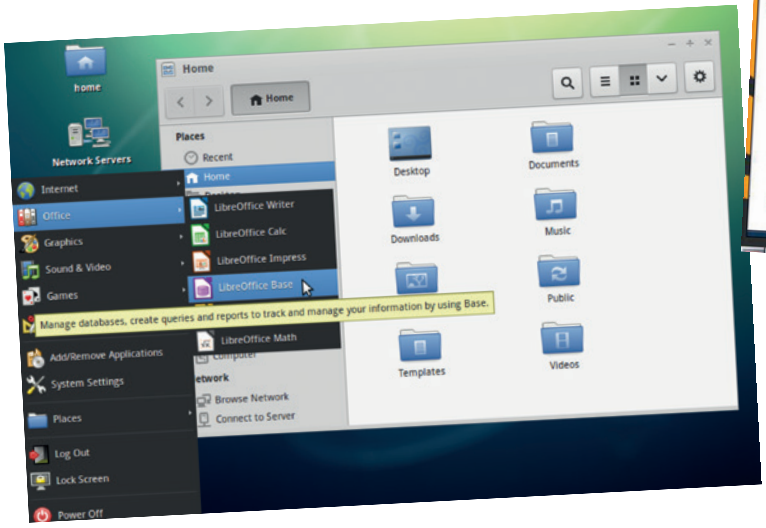
GNU/Linux distributions that are pure FOSS – every single byte.

Most big-name distros – Ubuntu, Fedora, OpenSUSE, Arch Linux, Debian – include some non-free software, or at least make such software easily accessible. There are a handful of distros that are completely Free Software, though, and which don't provide easy access to proprietary software, drivers or binary blobs.

That might seem overly restrictive, but the idea is to prevent you from accidentally installing something non-free. With one of these distros, you can be sure that your Linux installation is free to the core, regardless of whichever checkboxes you tick during the installation. Here's a selection of the most notable fully free distros:



Cinelerra: Program
File Edit Keyframes Audio Video Tracks Settings View Window
Video 1
Audio 1
Welcome to Cinelerra.
Napster Windows Help
Lopster 1.2.0 (0)
Server Chat Search Library
Name Status LineSpeed Users
WinMX Network
New Networks
Not connected
MuSE version 0.8.1 codename "ZOR2OR2"



Above: Dynebolic is geared towards multimedia, with audio and video editors making up the bulk of its software.
Left: Trisquel's default desktop is Gnome 3, but with the Classic mode enabled to make it behave more like Gnome 2.

Dynebolic

www.dyne.org/software/dynebolic

Created for "media activists, artists and creatives", Dynebolic GNU/Linux includes tools for recording, editing and encoding audio files, along with software for broadcasting the results on the net. It's designed to be lightweight, requiring just 256MB of RAM, and is therefore especially useful if you want to revive an old PC.

The developers are keen to state that the distro doesn't use any online cloud services. While Dynebolic has a heavy focus on media applications, it's also possible to install regular desktop software as well.

gNewSense

www.gnewsense.org

Despite the terrible name (seriously, try evangelising Free Software and telling them you use something called "guh-nuisance"), this distro is very polished with a well-presented website. Early releases of gNewSense were based on Ubuntu, but the most recent version uses Debian as its base and Gnome for the desktop. gNewSense is one of the few distros to achieve official support from the FSF – indeed, Richard Stallman used it for a while. The project has become somewhat dormant recently, though, with no major releases for almost two years.

Parabola GNU/Linux-libre

www.parabola.nu

Parabola is a fork of Arch Linux, but uses the Linux-libre kernel rather the standard kernel (so it has none of the aforementioned binary blobs). Additionally, it only provides access to fully Free Software in the Arch repositories, and maintains a blacklist of packages that don't provide the four freedoms (**http://tinyurl.com/nzdvsct**).

Like Arch, it's a rolling-release distro so it's being constantly updated – a bonus for those who like new software, but users need to keep an eye on the website for potentially distro-breaking changes that occur.

Trisquel GNU/Linux

http://trisquel.info

Our #1 pick of the fully free distros is Trisquel. It's based on Ubuntu and therefore inherits that distro's plus points: the user-friendly installer, the vast range of packages available in the repositories, and the wealth of help and support available on the net.

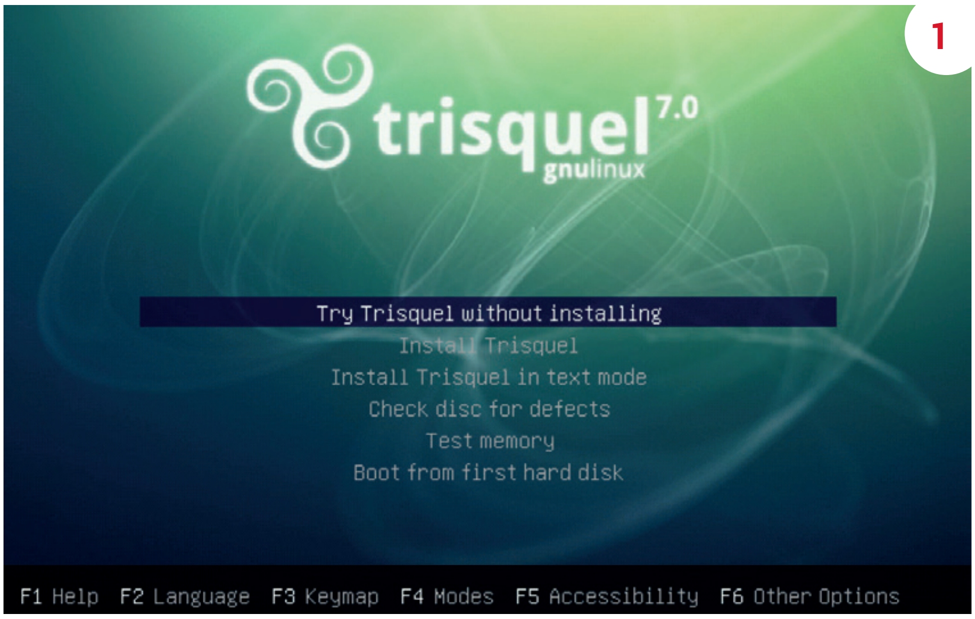
Trisquel's developers have focused on making the distro attractive and accessible, the end result being a fully free GNU/Linux flavour that you feel comfortable recommending even to newcomers. Its latest release is based on Ubuntu 14.04 LTS – so not bleeding-edge, but still reasonably modern.

INSTALLING TRISQUEL

Switching to a fully free distro couldn't be easier.

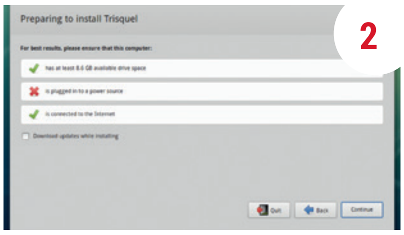
So, let's install Trisquel GNU/Linux. Go to **http://trisquel.info/en/download** and select the 1.5GB live DVD ISO image option. Scroll down and choose 64-bit or 32-bit (depending on your PC's CPU), then choose a mirror site close to you and click Download ISO. Recommended minimum system requirements are: a 1.8GHz CPU, 1GB of RAM and 10GB of hard drive space – but the more, the merrier!

Once you've downloaded the ISO image, you can burn it to a DVD-R and boot your computer from it (ie start your PC with the DVD in the drive), then follow the steps below to install Trisquel.



```
Try Trisquel without installing
Install Trisquel
Install Trisquel in text mode
Check disc for defects
Test memory
Boot from first hard disk

F1 Help  F2 Language  F3 Keymap  F4 Modes  F5 Accessibility  F6 Other Options
```

1 Boot At the initial boot screen, use the cursor keys to choose the Install Trisquel option and hit Enter. You can also check the DVD for defects, in case there was a problem when it was burnt.



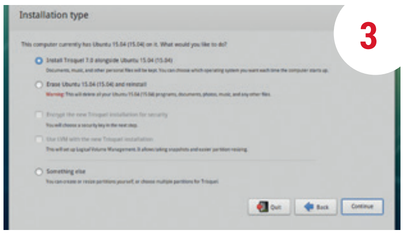
Preparing to install Trisquel

For best results, please ensure that this computer:

- ✓ has at least 8.6 GB available drive space
- ✗ is plugged in to a power source
- ✓ is connected to the Internet

☐ Download updates while installing

2 Prepare When the installer appears, it will check that you have enough hard drive space, that you have a power source and that you are connected to the internet.



Installation type

This computer currently has Ubuntu 15.04 (15.04) on it. What would you like to do?

● Install Trisquel 7.0 alongside Ubuntu 15.04 (15.04)
Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.

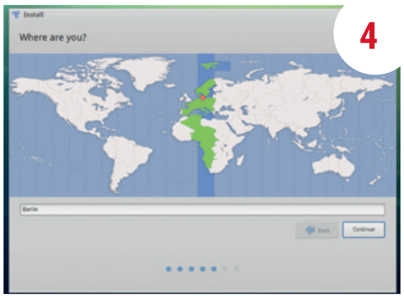
○ Erase Ubuntu 15.04 (15.04) and reinstall
Warning: This will delete all your Ubuntu 15.04 (15.04) programs, documents, photos, music, and any other files.

☐ Encrypt the new Trisquel installation for security
You will choose a security key in the next step.

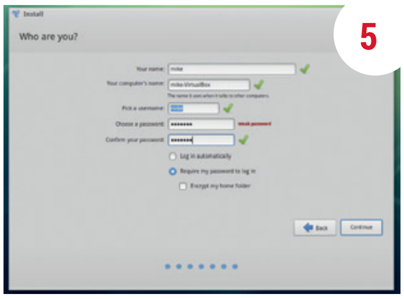
☐ Use LVM with the new Trisquel installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Trisquel.

3 Partition You can choose whether to install Trisquel alongside an existing Linux or Windows installation, or use the entire hard drive. If you're familiar with advanced partitioning, choose the Something Else option.



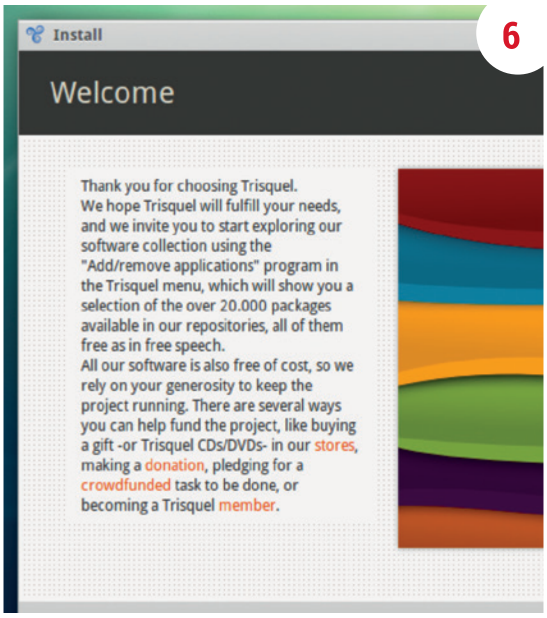
4 Timezone Next, you'll be asked to select your timezone and keyboard layout. Trisquel will try to guess where you are if you're connected to the internet – otherwise you'll have to choose manually. When you're done, click Continue.



Who are you?

Your name: mike ✓
Your computer's name: mike-VirtualBox ✓
The name it uses when it talks to other computers.
Pick a username: mike ✓
Choose a password: •••••••• Weak password
Confirm your password: •••••••• ✓
○ Log in automatically
● Require my password to log in
☐ Encrypt my home folder

5 User account In order to identify yourself to the Trisquel installation, you'll need to set up a user account and password. You can choose whether to log in automatically when you boot Linux, or ask it to prompt you.



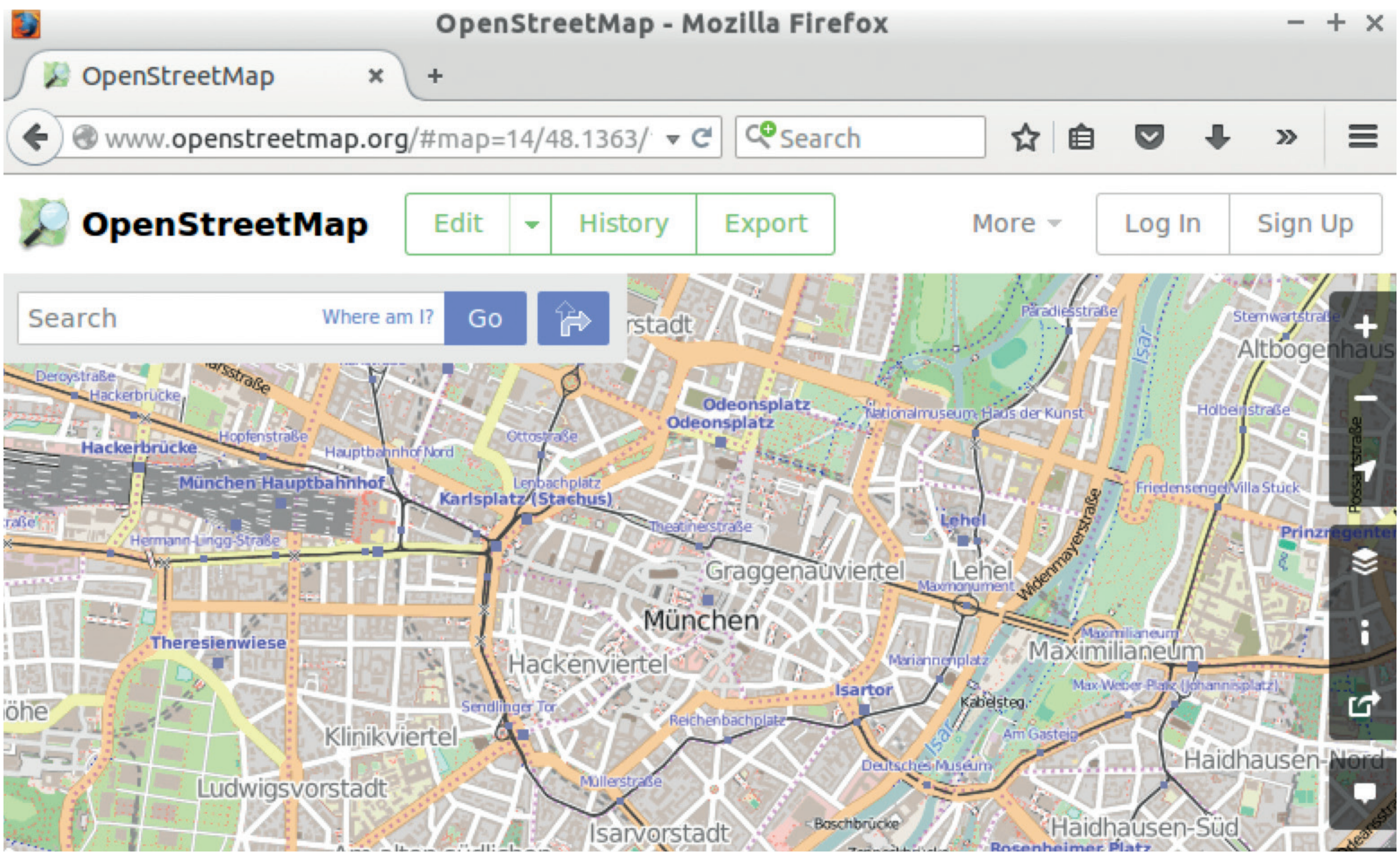
Welcome

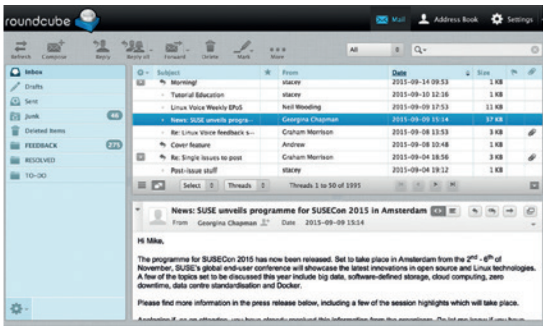
Thank you for choosing Trisquel.
We hope Trisquel will fulfill your needs, and we invite you to start exploring our software collection using the "Add/remove applications" program in the Trisquel menu, which will show you a selection of the over 20.000 packages available in our repositories, all of them free as in free speech.
All our software is also free of cost, so we rely on your generosity to keep the project running. There are several ways you can help fund the project, like buying a gift -or Trisquel CDs/DVDs- in our stores, making a donation, pledging for a crowdfunded task to be done, or becoming a Trisquel member.

6 Finish! After the files have been copied over, you can reboot your machine, remove the DVD or USB key, and boot up your shiny new Trisquel installation. Congratulations: you're now running 100% fully Free Software.

FREEDOM ONLINE

Switch to online services that put your privacy and security first.




roundcube

Mail | Address Book | Settings

Inbox, Drafts, Sent, Junk 46, Deleted Items, FEEDBACK 271, RESOLVED, TO-DO

Subject	From	Date	Size
Morning!	stacey	2015-09-14 09:53	1 KB
Tutorial Education	stacey	2015-09-10 12:16	1 KB
Linux Voice Weekly EPoS	Neil Wooding	2015-09-09 17:53	11 KB
News: SUSE unveils progra...	Georgina Chapman	2015-09-09 15:14	37 KB
Re: Linux Voice feedback s...	Graham Morrison	2015-09-08 13:53	3 KB
Cover feature	Andrew	2015-09-08 10:48	1 KB
Re: Single issues to post	Graham Morrison	2015-09-04 18:56	3 KB
Post-issue stuff	stacey	2015-09-04 19:12	1 KB

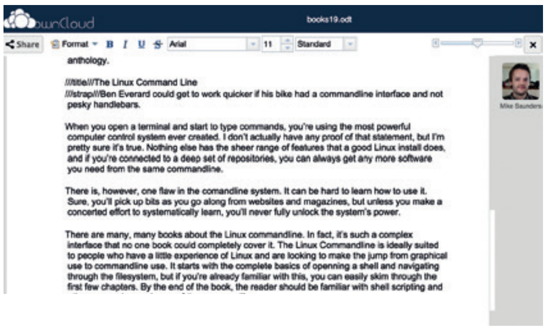
Threads 1 to 50 of 1995

News: SUSE unveils programme for SUSECon 2015 in Amsterdam
From Georgina Chapman Date 2015-09-09 15:14

Hi Mike,

The programme for SUSECon 2015 has now been released. Set to take place in Amsterdam from the 2nd - 6th of November, SUSE's global end-user conference will showcase the latest innovations in open source and Linux technologies. A few of the topics set to be discussed this year include big data, software-defined storage, cloud computing, zero downtime, data centre standardisation and Docker.

Please find more information in the press release below, including a few of the session highlights which will take place.


ownCloud books19.odt

Share | Format | B | I | U | S | Arial | 11 | Standard

anthology.

///title///The Linux Command Line
///strap///Ben Everard could get to work quicker if his bike had a commandline interface and not pesky handlebars.

When you open a terminal and start to type commands, you're using the most powerful computer control system ever created. I don't actually have any proof of that statement, but I'm pretty sure it's true. Nothing else has the sheer range of features that a good Linux install does, and if you're connected to a deep set of repositories, you can always get any more software you need from the same commandline.

There is, however, one flaw in the comandline system. It can be hard to learn how to use it. Sure, you'll pick up bits as you go along from websites and magazines, but unless you make a concerted effort to systematically learn, you'll never fully unlock the system's power.

There are many, many books about the Linux commandline. In fact, it's such a complex interface that no one book could completely cover it. The Linux Commandline is ideally suited to people who have a little experience of Linux and are looking to make the jump from graphical use to commandline use. It starts with the complete basics of openning a shell and navigating through the filesystem, but if you're already familiar with this, you can easily skim through the first few chapters. By the end of the book, the reader should be familiar with shell scripting and

Mike Saunders

Top: OpenStreetMap is much better than Google Maps in many respect: its interface is saner, and the maps more detailed. **Middle:** Most webmail services read your messages and pester you with ads. Get privacy by hosting your own with *Roundcube*. **Above:** OwnCloud features a rudimentary word processor, which is fine for basic tasks.

Even if all of the software installed locally on your machine is Free and open source, as soon as you start visiting websites, you run the risk of losing control again. Pretty much every website uses JavaScript now so it's easy to fall back into the trap of running code on your machine (in your browser) that you can't study, modify and share.

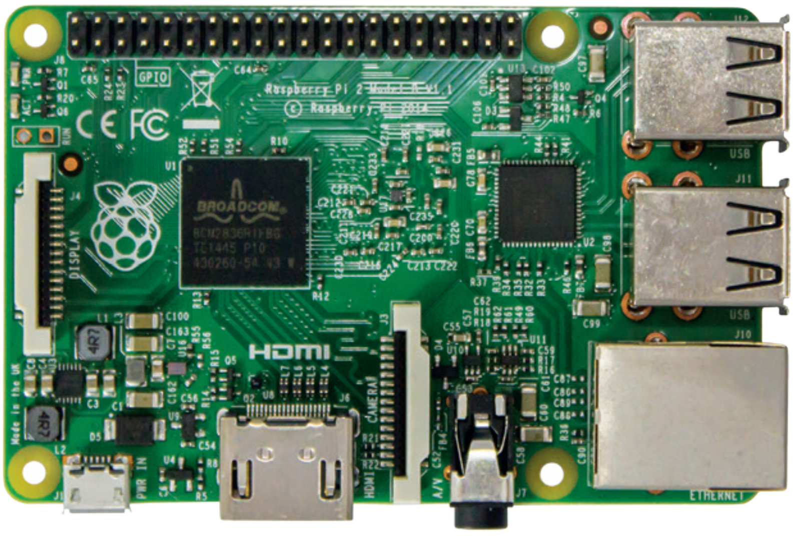
Now, even though the web is dominated by a handful of companies such as Google, Facebook and Twitter, it's perfectly possible to free yourself from their clutches. The first step is to install web browser add-ons that restrict the execution of JavaScript from a website until you explicitly allow it. *NoScript* (for Firefox) or *ScriptSafe* (for Chrome) let you enable JavaScript on an *ad-hoc* basis, so if a site doesn't need it and only uses it for fancy effects, you can disable it. In this way, you can stop third-party JavaScript from advertising providers from running on your machine. There's also *LibreJS* (**www.gnu.org/software/librejs**) from the free Software Foundation with similar goals.

Then it's worth installing a browser plugin that prevents websites from tracking you and following you around the web: *DoNotTrackMe*, *Blur* and *Privacy Badger* are good examples in this field. You may be tempted to install a full-on ad blocker, but many websites survive solely from advertising revenue, and we don't have anything against non-intrusive, non-tracking ads relevant to the site in question.

Search and social

Next up is switching to a FOSS-friendly search engine. DuckDuckGo (**www.duckduckgo.com**) is one of the better providers in this field – although not 100% FOSS on the back-end – with mostly decent results (if still lagging behind Google). It has a repo of source code at **https://github.com/duckduckgo**, and there's also a plain HTML version of the main search page, which doesn't use any JavaScript at **www.duckduckgo.com/html**.

Google Maps has a FOSS-friendly replacement in the form of


While the Raspberry Pi + Raspbian is mostly Free Software, the device uses non-free firmware to boot the GPU, so it's not the best choice if you want to go 100% free.

The colo alternative

If you choose commercial web hosting or a virtual private server, you may still end up running it on hardware that's not completely Free Software down to the core (ie it uses binary blobs or non-free firmware). If you really want to go 100% FOSS, you could always build your own machine and have it co-located in a hosting centre. These centres provide electricity, bandwidth, cooling and security for your machine, but otherwise it's yours to do with as you please – just as if it were in your home.

This can be quite expensive – upwards of £30 a month – and many co-location providers will only host machines in a specific form factor (eg rack unit servers). Others are more flexible, however, and for the right money they would be willing to look after your 100% Free Software GNU/Linux box. Some colo providers are now offering Raspberry Pi colocation, which is relatively cheap as they're small, they run cool and they use barely any power – but note that the Raspberry Pi isn't 100% FOSS and FSF-approved due to the firmware it uses.

OpenStreetMap (**www.openstreetmap.org**). This service uses data contributed from the community, and the database is free for everyone to share and modify. OpenStreetMap doesn't have some of Google Maps's frills, but it's often more up to date and has a much higher level of detail. Also, the OpenStreetMap data is used by many smartphone mapping apps such as the open source *OsmAnd*, which lets you download areas of the world for navigation when you have no internet connection.

Social networking

With every man Jack and his dog on Facebook now, it's hard for competitors to get a foot in. Diaspora's giving it a go though, with an open source and distributed social network, in that anyone can contribute a server (aka a "pod") and host accounts on it. (Contrast this to Facebook, which is massively centralised and entirely in the hands of a single company.) To find a pod that's open for new user accounts, see **https://podupti.me**.

For something more Twitter-esque, try Pump.io (**https://pumpit.info**), another distributed social network that's a continuation of the once mildly popular Identica network. Pump.io's userbase is very small in comparison to Twitter, but you will find some FOSS geeks on there and it has the potential to grow into something much bigger.

Hosting options

Of course, while using third-party services is one option, for maximum privacy you should host your own. This might sound like lot of work, but various packages are available that simplify the process considerably. First off, consider how you want to host your services: via your home internet connection, or somewhere else? The former is certainly possible, but most home broadband connections don't provide static IPs. In other words, your IP address will change every time the connection is dropped.

To counter this, you can use a DNS service like **http://freedns.afraid.org** – this regularly updates a domain name to point it at a frequently changing IP address. FreeDNS has a wide range of domains you can use, with custom subdomains, so you could use it to point **blah.homenet.org** to your home IP address. This is simple to set up, and involves running a program inside your home network at regular intervals (eg via **cron**) to make sure that FreeDNS knows your current IP address.

Hosting services from your home has downsides though: your upload speeds aren't likely to be amazing, which has a big impact if you want to create your own private file-hosting cloud, and many ISPs block ports for outgoing mail. In this case, you'll need a commercial hosting provider, and one of the best in terms of FOSS support is Gandi (**www.gandi.net**).

Gandi offers domain name registration, simple web hosting facilities, and VPSss (virtual private servers) for when you want maximum control over your OS and software. Once you have hosting or a VPS set up, you can install private cloud software such as OwnCloud (**www.owncloud.org**), which provides file storage, calendars and document collaboration. It's a quick way to replace Google Drive, Google Calendar and Google Docs (or their Microsoft equivalents) with services that fully respect your freedom. Gandi has useful documentation for setting up OwnCloud: **http://tinyurl.com/pjhg9lu**.

To replace Google Mail and co, you'll need to host your own mail server, a somewhat more involved task but certainly still possible. There are many different mail server packages out there, but we recommend using *Cyrus*, an IMAP server. We had a detailed tutorial on setting up *Cyrus* in issue 8 of Linux Voice, so grab a copy from **http://shop.linuxvoice.com**. Another piece of software worth considering is Roundcube, a slick webmail client that you can hook up to your own mail server and run on your hosting provider.

Even though the web is dominated by a handful of companies it's possible to free yourself from their clutches

ON TEST: LIBREBOOT X200

For maximum freedom, you want a machine that's FOSS right to the core.

The Libreboot X200 is a refurbished ThinkPad X200, and fully Free Software right down to the BIOS.

Other options

Currently, the Libreboot laptops from Minifree are the only machines that are certified by the FSF's Respects Your Freedom programme. So if you're looking for something else that's confirmed as running FOSS at every level, you're a bit out of luck. There's the Purism (**https://puri.sm**) range of laptops, which have similar goals, but as of the time of writing, they still included non-free software in the BIOS and some firmware, making them not eligible for Respect Your Freedom certification. If you're looking to build a PC from scratch, check out **www.h-node.org** for a database of hardware that works with fully free GNU/Linux distributions.

LINUXVOICE ON TEST

So you've installed Trisquel (or another fully free GNU/Linux distro), you've purged all traces of proprietary software from your machine, and you've switched to FOSS-friendly online services. What's your next step? Well, chances are that there's still some non-free software lurking in your machine. Most PCs and laptops have proprietary BIOSes (or UEFI implementations), for instance – and they are far from harmless. For instance, PC manufacturer Lenovo was recently caught forcing installation of bloatware on its Windows-based laptops via the machines' BIOS.

This didn't affect Linux users, of course, but it shows why it's important to have a fully open source BIOS if you want total control over your computer. Minifree (**www.minifree.org**) is a UK-based company selling Free Software Foundation-endorsed laptops that are FOSS right to the core, including the BIOS. The laptops are actually refurbished ThinkPads, primarily ex-corporate machines from large companies, and Minifree lent us one of its Libreboot X200 units (originally a ThinkPad X200) for review. This particular model is the highest spec that Minifree offers, with an Intel Core 2 Duo P8400 CPU clocking in at 2.26GHz, 8GB of RAM, a 240GB SSD drive and a docking station – all for €578, so around £433 at the time of writing.

> There's something very special about running a completely Free Software stack on such a robust, well designed machine

Business machine

Hardware-wise, the X200 is extremely rigid and well-built. It's a "business ultraportable" machine, and while its weight (1.66kg) and size is trounced by modern ultrabooks, it's still not especially heavy. The X200 has three USB 2.0 ports, VGA-out, SD card and ExpressCard slots, headphone and microphone jacks, plus ports for Ethernet, dialup modem and a Kensington lock. Its dimensions are 295mm (width) by 210mm (height) and its thickness varies from 21mm to 35mm. The docking station provides a CD/DVD-RW drive, four additional USB 2.0 ports and DisplayPort as well.

Along with the overall build quality, the biggest highlight of the hardware is

While the X200 isn't as thin as an ultrabook, it's still fairly light and has a superb keyboard.

the keyboard. It's absolutely fantastic, as fans of the older ThinkPad models will attest to.

Pointing fingers

No trackpad is installed in the X200; instead, you get a "nipple" TrackPoint device in the middle of the keyboard that can be used to push the pointer around the screen, along with three buttons underneath. A free USB mouse is available on request.

The X200's screen is perhaps the least impressive part of the package, and reflects on the machine's age. It's a 12.1-inch WXGA display (1280x800) that's lacking in terms of resolution, brightness and viewing angle when compared to more modern laptops. It's not bad per se – it just takes a while to adjust to and is a noticeable step backwards when you've gotten use to HiDPI displays. Some units have webcams, but not all – it's best to ask Minifree before ordering.

Now onto the biggest question: how well does GNU/Linux run on this machine? Fortunately, it's superb. Trisquel boots and shuts down very quickly thanks to the SSD, and Wi-Fi, volume controls and power management (eg suspend) work perfectly out of the box. This is quite an achievement for any laptop running Linux, but when you consider that it's 100% powered by Free Software, with no binary blobs in sight, it's even better. The X200 handles CPU load with aplomb too: even with maxed out CPU cores, the machine stayed cool and the fan noise was barely perceptible.

Nice little runner

In terms of performance, the 2008-era CPU and Intel 4500MHD graphics chip are a bit long in the tooth now, but they're no slouches and the 8GB of RAM – coupled with the SSD – make it a generally zippy machine to use. In our testing we found that daily tasks such as web browsing, image editing, office work and coding were no problem on the X200. Trisquel uses the GNU Flash Player (Gnash) to view Flash content on the web; this didn't work properly for many sites we tested, but YouTube ran smoothly using its HTML 5 mode. Battery life was decent as well, with the 9-cell battery providing around 4.5 hrs of usage with maximum screen brightness and light browsing and writing tasks. If you order an X200 and receive a 6-cell battery, expect around three hours on the road.

Overall, we can safely recommend the Libreboot X200 as a solid workhorse with superb GNU/Linux and FOSS support. Despite being refurbished, our review unit looked (and smelled!) largely like a new machine, and only a small trace of dust under the keyboard reminded us that it had a previous life in an office somewhere.

There's something very special about running a completely Free Software stack on such a robust, well designed machine – and with the perfect keyboard for heavy *Emacs* (sorry, *Vim*) sessions.

Verdict

A couple of slightly dated components aside, the X200 is a good value machine with superb build quality, almost flawless GNU/Linux support, and a heavenly keyboard.

★★★★☆

Minifree includes a power adaptor, Ethernet cable and FOSS stickers galore in the box.

OPEN RIGHTS GROUP

MEET THE ORGWELLIANS

Mayank Sharma embeds with the elite force that's guarding the UK's digital fence.

Like most good things that start with a question, it was a panel called "Where's the British EFF?" at the Open Tech Conference in 2005 that led to the creation of the Open Rights Group. The panel was hosted by Danny O'Brien, then the EFF's activist coordinator, and featured noted digital rights activists Cory Doctorow (representing the EFF), Ian Brown from European Digital Rights and Rufus Pollock of the Open Knowledge Foundation Network. An encouraging response from the attendees led O'Brien to set up a crowdfunding campaign on Pledgebank to secure enough funding to pay for a couple of members of staff. Around 600 people signed up, and ORG was born before the year was out.

The primary mandate of ORG is to preserve and promote your rights in this digital era. According to its website the group does "whatever it takes to build and support a movement for freedom in the digital age."

Wendy Grossman, who has been writing on computers and privacy issues since the early 90s, has been on ORG's advisory council since the beginning. She contacted O'Brien to get involved and he put her in touch with ORG's founding executive director, Suw Charman, when she began bootstrapping the organisation.

"ORG engages with many interests that in my case were fostered by attending the Computers, Freedom, and Privacy conference in the US every year from 1994 onwards. ORG engages with those same sorts of topics," explains Grossman in an email exchange. "It is much easier to shape the future of a medium at the beginning than it ever will be later on when interests (and sometimes bad laws) have become entrenched. The internet is the one new medium of my lifetime, and how we shape it will set the tone for generations to come. Bad policy decisions now will weigh on our great-grandchildren."

The Open Rights Group focuses on the intersection between technology and human rights

An ORGan for change

In an email exchange with LV, Pam Cowburn, ORG's communications director, notes that it campaigns for both free speech and privacy online: "We focus on the intersection between technology and human rights. ORG believes people have the right to control their technology, and oppose the use of technology to control people." She reasons that given how much technology has transformed the way we communicate, work and live, this is a massive remit, so the group has to be selective in its approach. The group focuses on the most important issues that they can influence before deciding the most effective way to bring about change. ORG also works in partnership with other organisations, as it helps them be more effective with their limited resources.

Cowburn explains that a lot of their work involves reacting to external developments: "Our supporters rely on us to respond to the behaviour of governments, corporations and others. Following the Snowden revelations, we have inevitably focussed on surveillance by the British and other governments.

This campaign has involved challenging government legislation such as the Data Retention and Investigatory Powers Act (DRIPA) and working with other organisations to call for more transparency, accountability and better oversight. With other partners, we're challenging the UK Government at the European Court of Human Rights." ORG also intervened in a case brought by MPs Tom Watson and David Davis, which saw the High Court rule that parts of the Data Retention and Investigatory Powers Act were unlawful.

Quis custodiet ipsos custodes?

The group has produced in-depth papers on a wide variety of subjects. There's a report detailing GCHQ's mass surveillance programs, the intrusion and integration with the NSA, and the threats and risks it creates. Another one highlights the group's digital rights concerns with the Transatlantic Trade and Investment Partnership (TTIP). You can also read its correspondences with the government and public functionaries on issues such as the exchanges with London's Police Commander Steve Head on the Police Intellectual Property Crime Unit (PIPCU).

Outlining the group's upcoming tasks, Cowburn shares ORG's concerns over the new surveillance bill on the horizon: "We are concerned that the government will attempt to extend its powers and bring back elements of the draft Communications Data Bill, which ORG and our supporters helped to defeat previously."

Commenting on their course of action in case this happens, she says the group will mobilise its supporters and ask them to persuade their MPs that human rights do matter: "Our supporters are key to this campaign. We know that MPs listen to their constituents, especially when they care enough about an issue to visit them in person. We've already held a lobby day in parliament to help people set up meetings with their MPs. Once the new draft bill is published, we'll no doubt be asking supporters to raise concerns with their MPs whether through emails or face-to-face meetings."

The ORGanisers

The Open Rights Group has a handful of paid staff members, headed by executive director Jim Killock. Before joining ORG in 2009, Killock worked as the external communications co-ordinator of the Green Party. A grassroots man himself, Killock had led campaigns against the 'three strikes' and the Digital Economy Act, the company Phorm and its plans to snoop on UK users, and against pervasive government internet surveillance.

Another member of staff is project manager Richard King, who oversees technical projects. While ORG might appear to be a campaigning organisation, King reasons that because ORG campaigns at the intersection of human rights and technology, "it's natural for us to use technology in our campaigns."

Credit: Jim Killock

Credit: Sheila Thomson

Bristol Festival of Ideas
In association with The Observer University of Bristol
Cory Doctorow
Information Doesn't Want to be Free
Chair: Dick Penny
www.ideasfestival.co.uk
@Festivalofideas #bfoi
Credit: Ed Lander

Top: Project Manager Richard King with ORG supporters at an ORG hack day at Mozilla's offices. **Middle:** Glyn Wintle, an ORG supporter from the very beginning, mans the group's stall at LUG Radio Live. **Above:** Cory Doctorow interacting with attendees at an event organised by ORG Bristol.

Top: Supporters of the Open Rights Group protest against the Digital Economy Bill back in 2010. **Above:** A section of the crowd at ORGCon 2012.

King coordinates the work of ORG's community of technical volunteers, which is made up of ORG supporters who like to work on digital-rights projects they think are interesting. "We're not just a community of coders," he explains. "We encourage people with any kind of interest in technology to join in. We especially love it when people suggest new projects – and it's my job to support these in whatever way I can. I also organise occasional hack-days for the community." King points to **https://www.blocked.org.uk** as example of a project built by the tech-volunteers. The website makes the web filters of home and mobile ISPs transparent for the first time. The system is Free Software and the group is now working to deploy it in other countries as well.

ORG also taps the collective knowledge of its advisory council for guidance and expertise. Cowburn points out that the advisory council is made up of tech experts, MPs, academics and activists. "They have vast amounts of knowledge, expertise and experience and we call on them regularly to ask their advice about our strategy and campaigns, to comment on our reports or to help us to give expert responses to media requests." It has over 40 members, including John Buckman (founder of Magnatune), Alan Cox (noted kernel hacker), Graham Linehan (writer and director of *The IT Crowd*), Cory Doctorow, Wendy Grossman and others.

Cowburn also points out that ORG also has a group of lawyers to whom they look for advice and guidance when required. She believes "legal interventions are a very effective way of campaigning for change" and highlights the work of David Allen Green, a lawyer at Preiskel & Co LLP, who acted for ORG *pro bono* in a trademark infringement case.

If we allow the internet to be taken over by those who don't understand its importance, the impact would be huge

Circle of friends

An important proponent of ORG's outreach to individuals are the different local groups all over the UK. "Our local groups are really vital to ORG's success," says Cowburn. "They are a place where supporters can meet and get together to discuss digital rights issues. Like many organisations, we are based in London and it's easy for organisations to become very London-centric. Our local groups help us to reach out to people across the country and are invaluable in putting pressure on their local MPs."

Each local group is managed by local representatives or organisers, and ORG provides information and support to help anyone new to campaigning. The group organisers are assisted by the Local Groups Co-ordinator and ORG staffer Lydia Snodin, who helps the organisers with planning, administrative and perhaps even some financial support. The group organisers are invited to join the Supporter Council, where they can share ideas to build their groups, brainstorm event ideas and help plan future campaigns.

Art O Cathain is a software developer and local organiser of the Bristol group and has planned and hosted several ORG events in Bristol. "Organising events locally is a case of coordinating a speaker, venue, and audience. It's useful to have the Open Rights Group name because they are a well-known organisation within the tech community. For example, we recently held a talk by Cory Doctorow during the Bristol Festival of Ideas. ORG's campaigns coordinator helps out with suggesting speakers and event themes, and publicising events."

The local organiser of the Manchester group, Tom Chiverton, narrates a similar tale: "The campaigns team at HQ always seems to be available for last-minute thoughts and sharing of ideas." In addition to finding speakers, Chiverton says that ORG also helps fund larger events, as well as travel costs for the more far flung speakers.

Get involved

If ORG's activities resonate with you, the group presents several opportunities for contributions. While one of the best ways to contribute is to find and join a local group, there are other avenues. King invites all

those who are technically inclined to contribute to ORG's tech efforts by joining its mailing list at **https://lists.openrightsgroup.org/mailman/listinfo/tech-volunteers**. Those with an editorial bent of mind can contribute to the internal research projects about projects that interest ORG.

ORG keeps track of developments in the UK parliament that may have an impact on digital rights issues such as copyright, privacy, and open data. When it finds something that needs to be addressed, the ORG seeks opinions and inputs for drafting consultations for the Government. A consultation is a government request for opinions on various issues, be they proposed internal guidelines, draft legislation, or simply the route the government wants to take in a particular area.

One sub-section on ORG's wiki is the Adopt an MP scheme. This page hosts a list of the Members of Parliament along with their contact details. You can read the views of the individual members on the various digital rights issues. Volunteers can update the page on the MP to include their public statements on issues that are of interest to ORG. This list helps when the groups needs to lobby MPs.

If you can earmark specific periods of time for helping out ORG, you can join the group as a volunteer working on the things mentioned on the volunteer page (**https://www.openrightsgroup.org/volunteer**). The group also offers some internship positions for students during their summer break. If this interests you, check out open positions at the internships page on the wiki (**https://wiki.openrightsgroup.org/wiki/ORG_Wiki:Internships**).

The Open Rights Group is funded by small grants and donations from individual supporters. You can become a member by giving as little as £2.50 a month by Direct Debit, although the usual amount varies between £5–10 a month. Members who pay £60 a year or more can opt to receive a welcome pack and gift, which currently is a copy of Becky Hogge's *A Guide to the Internet for Human Rights Defenders*. You can also make a one-time donation via PayPal and the group even accepts Bitcoins.

Defend the future!

The issue of digital rights has never been more relevant in the UK than it is now. As companies and the government come up with more shrewd technology and laws that will invariably intrude upon our privacy, we need someone who can look after the interests of individuals. Navigating the digital minefield requires knowledge of technology, policy and law and most of us are ill-equipped to grasp the implications.

Cathain says he is a passionate believer in freedom and liberty, which is why he thinks the Open Rights Group is so important. "Because computers and technology are generally not well understood, only a small fraction of the population appreciates the significance of digital rights issues. The same problem applies to our elected MPs. Laws haven't kept up with changes in technology and it's vital that they can debate new laws such as the proposed Snoopers' Charter with the necessary understanding of the issues."

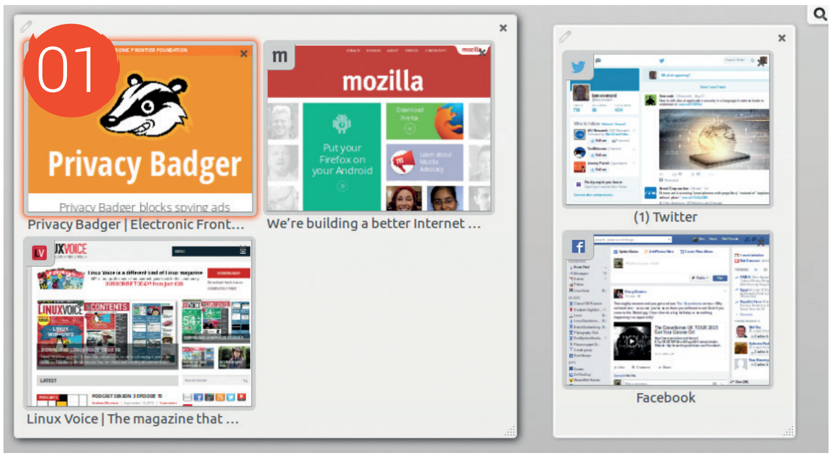
Chiverton holds a similar view: "The online world is no longer an optional part of modern life, and a functioning democracy requires a free flow of ideas and information. If we allow the internet to be taken over by those who don't seem to understand its essential character and importance, and would seek to turn it into a series of silos and blocks of bunkers, the impact would be huge."

Above: ORG is using £1,000 donated by Linux Voice to campaign about surveillance in the run up to the publication of the draft Investigatory Powers Bill this autumn.

SECRETS OF FIREFOX

Master the web with the hidden features of the flagship open source browser.

If there's one project that has brought more new users to open source software than any other, it's *Firefox*. This web browser rose out of the now-defunct Mozilla suite and came to dominate the web, though it now faces stiffer competition as other new web browsers have come along and challenged for online supremacy. Here at Linux Voice, we think that *Firefox* is still the best web browser around, but takes a bit of experience to get the most out of it. Here are some of the features that we find particularly useful...


01
Privacy Badger
Privacy Badger blocks spying ads
Privacy Badger | Electronic Front...
mozilla
We're building a better Internet ...
(1) Twitter
Linux Voice | The magazine that ...
Facebook

02
Privacy Badger | Electronic Frontier Foundation - Mozilla Firefox
A PROJECT OF THE ELECTRONIC FRONTIER FOUNDATION
Privacy Badger
Privacy Badger blocks spying ads

01 Tab control

Most *Firefox* users are comfortable using tabs to keep multiple websites open at a time, but *Firefox* tabs are more powerful than this. Pinning tabs (right-click on the tab and select Pin Tab) keeps the tab on the left-hand side and the browser opens that site automatically when launched. If you use large numbers of tabs, you may find it easier to keep track of all the pages by grouping tabs. Press Ctrl+Shift+E to enter the Tab Group screen to ease tab management.

02 Privacy

There are a wide range of options to increase your privacy when using *Firefox*. Disabling third-party cookies (Edit > Preferences > Privacy > History > Use Custom Settings For History > Accept Third Party Cookies > Never) is a simple way of cutting down the number of sites that track your browsing. If you want to go further, addons like *Privacy Badger* (from the EFF) and *NoScript* enable you to make intelligent choices about how you want your browser to work.

03 Developer tools

These enable you to get a better insight into how the browser's working – for instance, you can use them to analyse a web page's performance, debug its CSS and inspect the HTML. Whether you're building a website or investigating someone else's, there's no better tool for understanding how the web works.

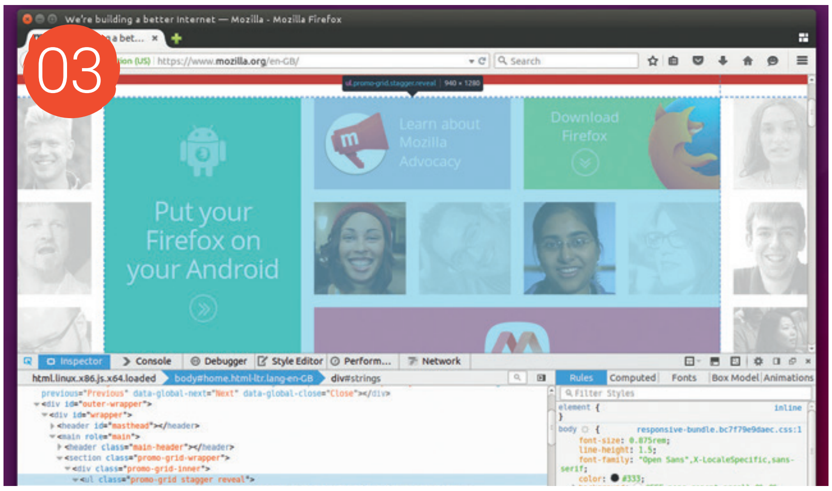
04 Addons

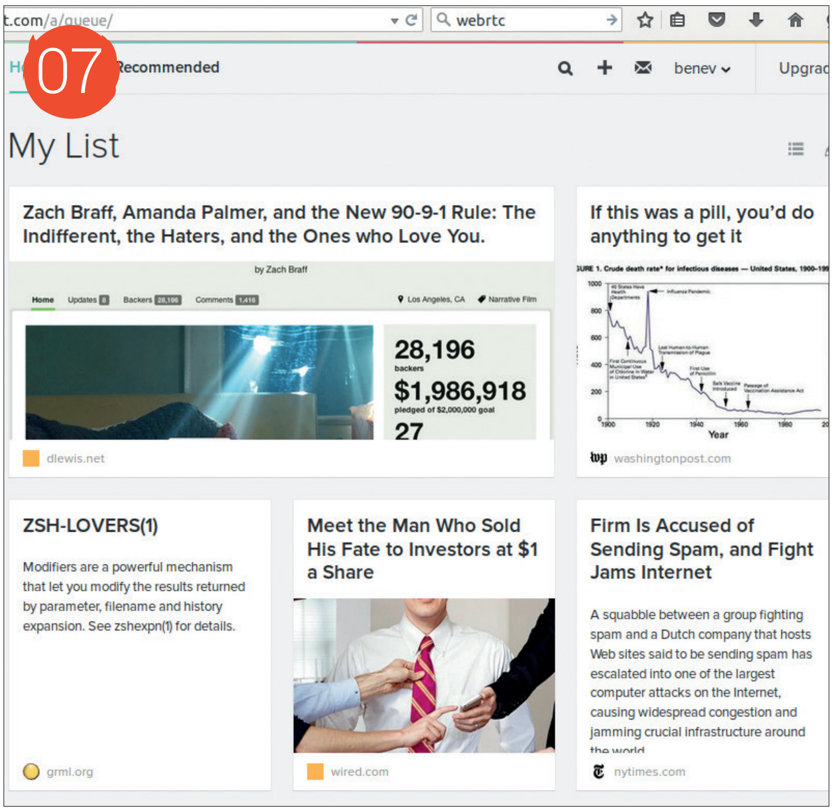
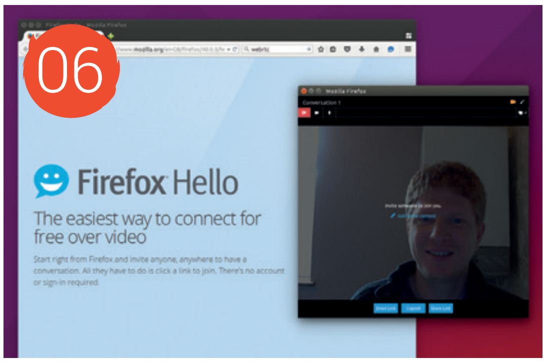
Firefox gets a few more features every release, but it'll never have everything that everyone wants. However, it does enable you to get the features you're missing by using add-ons. Head to **https://addons.mozilla.org** to see what's available.

05 Themes

If the default grey is a little boring for you, you can spice up your browser's look with a new theme. These are a special type of addon that just affects the look of the browser, leaving its features unchanged. You can take a look at the options (be warned, some of them aren't too pretty) at **https://addons.mozilla.org/en-/firefox/themes**.

Addons like Privacy Badger (from the EFF) and NoScript enable you to make intelligent choices about how you want your browser to work



06 WebRTC

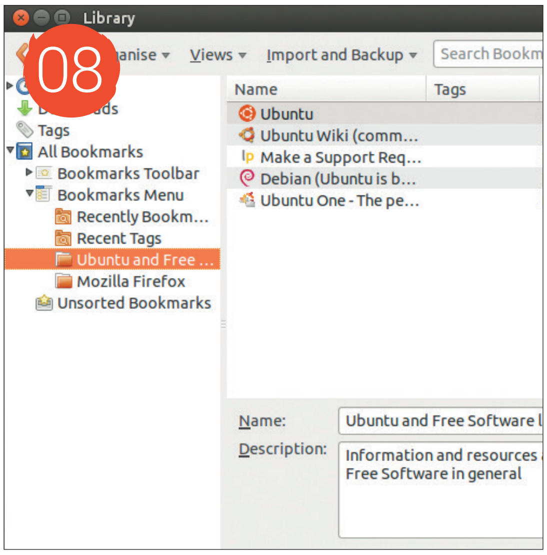
This black magic can send data directly between web browsers. At the time of writing, WebRTC is mostly used for video and text chat that doesn't rely on central servers, but the technology is data-agnostic, so it can be used to shuffle anything between browsers in real time.

07 Pocket

This is a mobile app that enables you to save web pages for offline viewing. If you install *Pocket* on your mobile or tablet, then you can press the Pocket button in *Firefox* on your desktop on a page you want to save. Your portable device will then download the page and make it available offline.

08 Bookmark management

Visit the same pages often? Bookmarks make it easy to go back without having to remember the address. You can also make it easy to manage your bookmarks by grouping them into folders. To enable easy access to your bookmarks, you can add a toolbar to the main *Firefox* window (View > Toolbars > Bookmarks Toolbar) or adding them to the new tab page (use the pin icon when they appear). Happy browsing!



ANDROID ANATOMY

Take control of the most widely distributed Linux-based computing device on the planet.

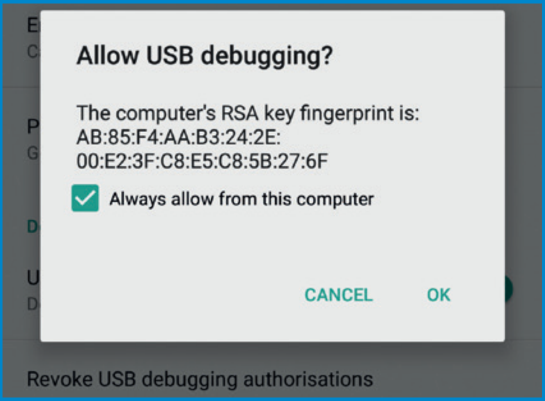
It may be just a phone. Or a tablet. And it may be just a simple, perfunctory, useful tool; the prerequisite accessory of our times. But our phones hold many times the power and capacity of even PCs from just a decade ago, and they're always connected, always talking to servers, masts and satellites. And most remarkably of all, this frontier is mostly running on Linux.

Beneath the surface, clinging to the beating heart of its Linux kernel, there's still a lot that will feel familiar in Android. Security via SELinux plays a central role, for example, and the filesystem has the same general layout as your desktop.

But the smartphone is also the new proprietary frontier. Where once we railed against PCs pre-sold with a closed operating system that couldn't be changed, we find ourselves in even greater peril. These are PCs in our pocket, continually connected and capable of running software 24 hours a day, 7 days a week, and yet most are locked down in ways many wouldn't accept from PCs. They're what computing would have looked like had UEFI and SecureBoot realised our worst nightmares.

But there is hope. Android is open source, and the first step to a better platform is understanding how it works, and how you can use your Linux skills to subvert and extend this computer in your pocket. This is what we're going to do over the next few pages, giving you the confidence to do anything from make a few changes to your phone, to completely replacing the operating system.

DANGER!
Messing around with elements of your Android devices can cause them to break, void warranties and become worthless. Proceed with extreme caution, or at the very least, proceed only after you've copied and backed up your data.


Allow USB debugging?

The computer's RSA key fingerprint is:
AB:85:F4:AA:B3:24:2E:
00:E2:3F:C8:E5:C8:5B:27:6F

✓ Always allow from this computer

ADB needs to be enabled as 'USB debugging' in your phone's system configuration window.

THE FILESYSTEM

Remember: Everything is a file

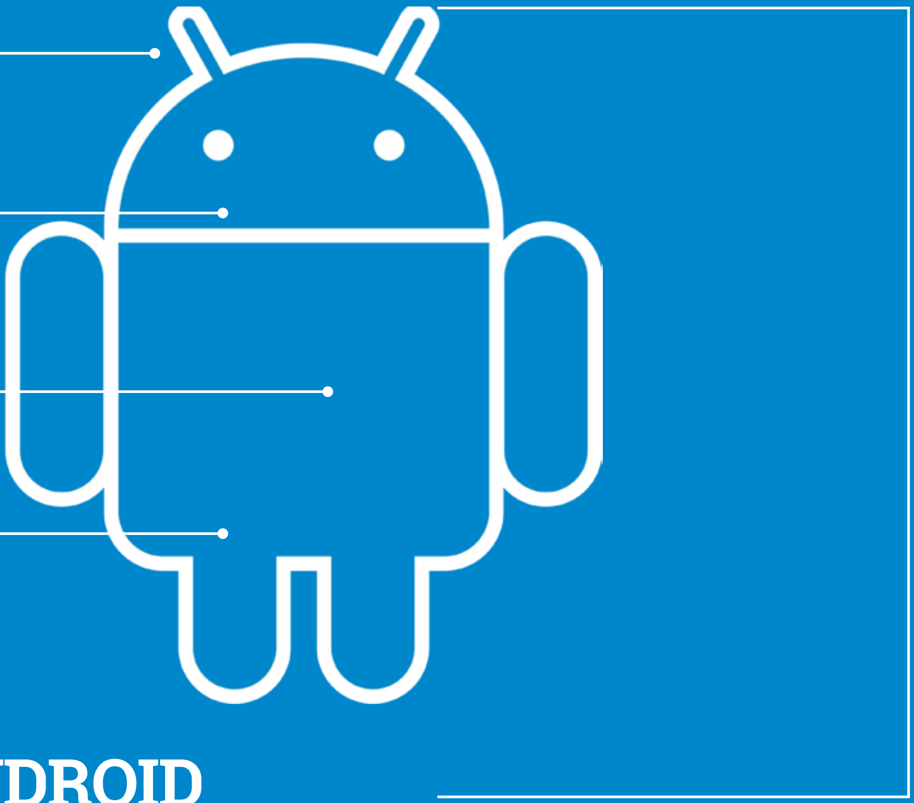
The key to understanding the similarities and the differences between Android and your typical Linux installation is to understand the Android filesystem and how storage is partitioned. Each Android device is different, so the best way of seeing how your phone is configured is to check for yourself from the command line. On the vast majority of Android-based phones, you can install a terminal emulator and use the command line just as you would your Linux box, and there are various terminal applications available in both the Google Play store and the open source F-Droid store. Simply running the terminal and typing **cat /proc/partitions** will return a long list of devices that all start with 'mmc', which originally

Memory allocation

Android's greatest/most frustrating attribute is that every device is different, and almost every manufacturer does things slightly differently from other manufacturers from one generation to the next of each device. The most open and experimental Android devices we've found are Google's own Nexus units, and these have become the closest thing to a standard reference point, whether or not other manufactures take any notice. Nexus devices have relatively little obfuscation of the operating system, the bootloader is open, and it's relatively easy to get yourself root access (a process known as 'rooting' on many devices). You'll also find that lots of other people have been doing the same thing, so you can often find support or details of what's possible, and there's a proliferation of third-party operating systems to install.

Canonical's Ubuntu Phone operating system is based on Android and embeds significant chunks within Ubuntu Touch, which means that much of the 'under-the-hood' Android specification is also applicable to an Ubuntu Phone. The partitioning scheme is similar, for example, although the 'recovery' partition currently doesn't do anything. Fastboot works, however, and the recovery partition can still be flashed with an Android equivalent for the same phone, as can the entire phone, and ADB can be enabled by activating 'Developer Mode' in the 'System Settings > About This Phone' pane.

Google has just announced its Nexus 5x and 6P phones, but the original Nexus 5 is only marginally less powerful and still a great option for hacking.


OPEN DEVELOPMENT PLATFORM – NO WALLED GARDEN
PSEUDO-ANARCHIC SYSTEM LAYOUT
LINUX KERNEL HEART FOR FLEXIBILITY AND STABILITY
UPGRADE THE OS WITHOUT WAITING FOR YOUR PHONE CARRIER

KNOW YOUR ANDROID

stood for 'Multi-media card' when flash memory was removable. The chips are now more usually fused onto the PCB of modern models and split into 512-byte sectors, much like hard drives. Our Nexus 5, for example, listed **mmcblk0Op1** through to **mmcblk0Op29**, plus a partition called **mmcblk0rpmb**.

To see where these partitions are being used, you can use the same **mount** command you would on your desktop, and you should pipe the output through **grep**, searching for 'msm,' to avoid being inundated with text:

```
1|u0_a184@hammerhead:/ $ mount | grep msm
/dev/block/platform/msm_sdcc.1/by-name/system /system ext4
/dev/block/platform/msm_sdcc.1/by-name/userdata /data ext4
/dev/block/platform/msm_sdcc.1/by-name/cache /cache ext4
/dev/block/platform/msm_sdcc.1/by-name/persist /persist ext4
/dev/block/platform/msm_sdcc.1/by-name/modem /firmware
vfat
```

The device nodes in the above output match the partitions that can be found in **/dev/block/platform/msm_sdcc.1**, although the contents of these device nodes are hidden unless you gave root access (see later for details on how to get root), and these match against the partitions we listed earlier. MMC flash devices appear to the system as hard drives, they're formatted using Android's preferred filesystem, which is currently ext4, as shown above. As these are the partitions used while the operating system is running, there are a couple more that aren't mounted and aren't visible from within Android itself. The first is **boot**, which performs exactly the same function as **/boot** in GNU/Linux, and **recovery**, which is a special partition used to update and recover your phone.

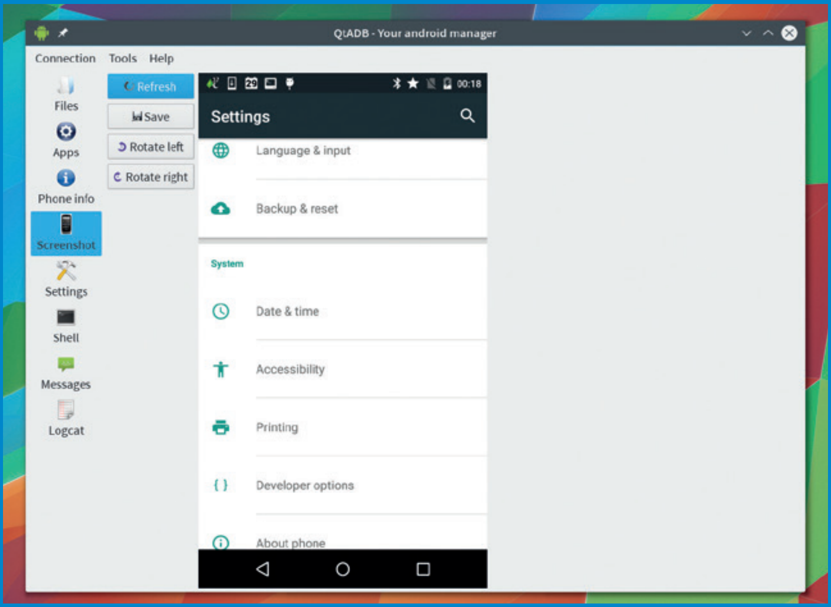
Main partitions

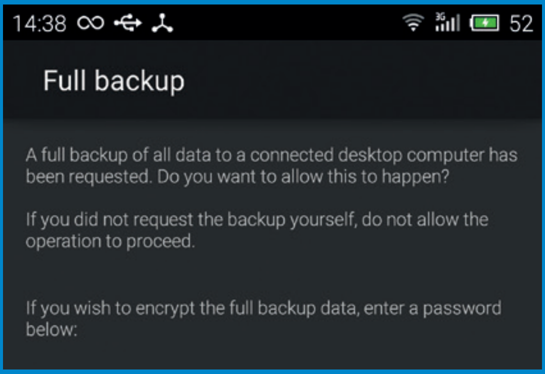
- **boot** The **boot** partition performs exactly the same function as **/boot** on the Linux desktop, but it doesn't contain the bootloader – that's on a different part of the flash memory. **boot** contains the kernel and the ramdisk for the operating system, both of which are read almost as soon as the system is turned on.

The ramdisk is a minimal root filesystem that's loaded into memory and used to boot the remainder of the system, and becomes the root filesystem of the operating system via the **rootfs** mount point after booting. The init scripts, found in **/init**, are then run to boot the remainder of the system. The kernel version also needs to correspond with the drivers contained in the system, or a mismatch may prevent Wi-Fi or Bluetooth from working, or even a successful boot.

- **system** Here's where all the main Android components of the operating system are installed. Before Android 5.0, it used to correspond directly to the contents of the **system.img** file that was part of any Android update or installation, but it's now rolled into system.new.dat which needs to be opened and re-created using a specific tool.
- As the **mount** command reveals, it's also mounted read-only so that the user can't mess around or alter its contents. Removing and replacing files in

Android is open source, and you can use your Linux skills to subvert and extend the computer in your pocket





Full backup

A full backup of all data to a connected desktop computer has been requested. Do you want to allow this to happen?

If you did not request the backup yourself, do not allow the operation to proceed.

If you wish to encrypt the full backup data, enter a password below:

Above: Backing up through *ADB* is the best way we've found to keep your Android device safe, especially as ADB can also work across Wi-Fi.
Right: With *ADB* running, you don't even need to use the command line. The brilliant *QtADB* lets you perform lots of advanced functions on your phone with just a mouse.

the system partition, often referred to as a device's ROM, especially for devices that ship with their own front-ends, is what the modding community spends a great deal of time doing.

- **recovery** This partition loosely corresponds to the rescue partitions you find on laptops. It's usually booted into by holding down a button combination when you turn your phone on – most commonly power and volume-down at the same time, and it usually offers a way to restore the operating system if it's become corrupt or broken. But what the partition contains depends on the manufacturer. On many devices, for example, after booting into the recovery partition you can mount the partition on your computer and copy across an update/install file for Android. The recovery partition is also used to store and the ClockworkMod (CWM) or TWRP boot selectors/backup tools, as they can be booted to with the recovery shortcut.
- **data** This will appear as **userdata** in the **mount** command. This is where all those applications you download are installed, keeping the system partition untainted. This is also why you'll find this partition mounted as read/write, although only by 'root,' which you'll need to be if you want to look around. If you need to back up your data, such as messages, emails and photos, this is the partition you should focus on, as it's also the partition that's wiped by a factory reset. data is related to the **cache** partition, which stores temporary files that need to be accessed frequently.

ADB

One of the easiest ways to access your phone's internals is through the *Android Debug Bridge*, *ADB*. *ADB* enables you to open a shell on your Android device from a computer connected with USB, much like connecting to a remote SSH server through a network. This configuration means there are two sides to this configuration. On the Android device, you need to be able to activate its 'debug mode.' On many Android devices, including Google's Nexus 5 and Samsung Galaxy phones, this is accomplished by tapping 7 times on the 'Build Number' field in the 'About phone' page, unlocking the Developer Options settings pane. Other phones, such as those running Meizu's Flyme OS, include the option as a simple switch in the settings panel.

Before you can connect your Android device to your computer, you will need to install the *ADB* drivers and do a little pre-configuration to make the connection work. *ADB* is part of the Android development kit, which means if you've already done some Android programming on your computer (such as following our short series on Android app development in LV004), you won't need to install anything else. Otherwise, you'll need to either install the complete SDK or a cut-down version of the tools package, if your distribution provides one.

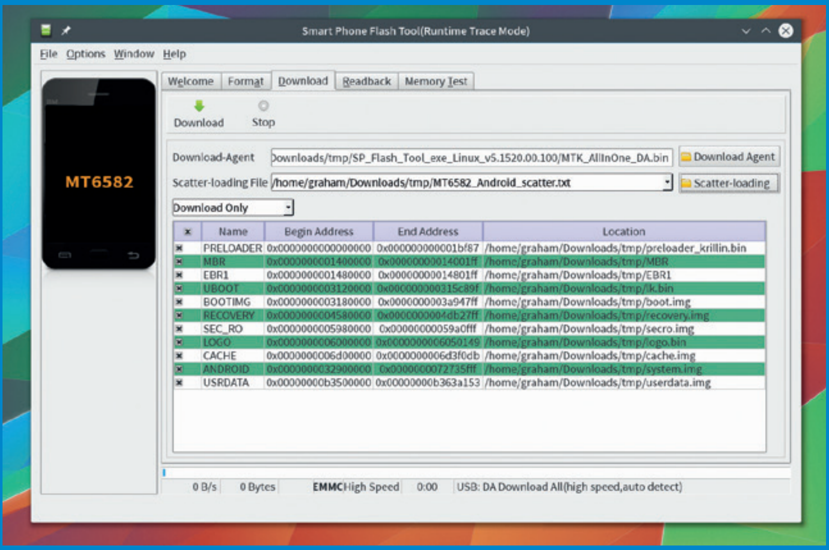
You can find these by searching for **adb** or **fastboot** specifically (Ubuntu's packages are **android-tools-adb** and **android-tools-fastbootm**, for instance). With everything installed, we need to explicitly state which device we want to speak to. With the phone connected, type **lsusb** and look for something similar to the following:

```
Bus 001 Device 008: ID 18d1:4ee1 Google Inc. Nexus 4 / 10
```

You need to add the Vendor ID (the **18d1** for our Nexus 5 above) to the file called **adb_usb.ini**, which should be within a **.android** folder in your home directory. As long as **.android** exists, you can do this by typing **echo 0x18d1 >> ~/.android/adb_usb.ini**, but you'll need to change the Vendor ID for the one returned by your phone. Now connect your phone and type **adb devices**, which is a command that will list *ADB* compatible devices.

ADB is a daemon that runs in the background, so this will need to start first. After a few moments, you'll see a prompt on your phone to accept the connection from the computer and, after doing this, the output from *ADB* will show your device is connected.

All the exciting stuff happens with Fastboot, which is most commonly used to reflash Android's partitions


Smart Phone Flash Tool(Runtime Trace Mode)
File Options Window Help
Welcome Format Download Readback Memory Test
Download
Stop
MT6582
Download Agent
Scatter-loading

Download-Agent: Downloads/tmp/SP_Flash_Tool_exe_Linux_v5.1520.00.100/MTK_AllInOne_DA.bin
Scatter-loading File: /home/graham/Downloads/tmp/MT6582_Android_scatter.txt
Download Only

✖	Name	Begin Address	End Address	Location
✖	PRELOADER	0x0000000000000000	0x000000000001bf87	/home/graham/Downloads/tmp/preloader_krillin.bin
✖	MBR	0x0000000001400000	0x00000000014001ff	/home/graham/Downloads/tmp/MBR
✖	EBR1	0x0000000001480000	0x00000000014801ff	/home/graham/Downloads/tmp/EBR1
✖	UBOOT	0x0000000003120000	0x000000000315c89f	/home/graham/Downloads/tmp/lk.bin
✖	BOOTIMG	0x0000000003180000	0x0000000003a947ff	/home/graham/Downloads/tmp/boot.img
✖	RECOVERY	0x0000000004580000	0x0000000004db27ff	/home/graham/Downloads/tmp/recovery.img
✖	SEC_RO	0x0000000005980000	0x00000000059a0fff	/home/graham/Downloads/tmp/secro.img
✖	LOGO	0x0000000006000000	0x0000000006050149	/home/graham/Downloads/tmp/logo.bin
✖	CACHE	0x0000000006d00000	0x0000000006d3f0db	/home/graham/Downloads/tmp/cache.img
✖	ANDROID	0x0000000032900000	0x0000000072735fff	/home/graham/Downloads/tmp/system.img
✖	USRDATA	0x00000000b3500000	0x00000000b363a153	/home/graham/Downloads/tmp/userdata.img

0 B/s 0 Bytes EMMC High Speed 0:00 USB: DA Download All(high speed,auto detect)

If the worst happens, there's a good chance you can even fix a phone that doesn't turn on with *SmartPhone Flash Tool* – but it won't restore the vital number for your device.

To open a shell that's running on your device, type **adb shell**. Just like running the Terminal app on the phone itself, you'll be able to run commands and explore the filesystem. You can copy files to and from your device using the **adb pull** and **adb push** commands, with full path names for source and destination as their arguments. You can also install and uninstall packages with the **install** and **uninstall** arguments, which is an essential utility if your phone can't connect to a network and doesn't have any removable storage. Anther great command is **adb logcat**. This will output the system log of your device, and continue to show new entries as they occur, which is brilliant if you're trying to troubleshoot some element of your installation. Then there's perhaps the most essential:

```
adb backup -apk -shared -all -f backup.ab
```

This will back up all your self-installed applications and the contents of your shared storage, putting everything into the **backup.ab** file. When entering this command, you'll have to accept the request on your phone, optionally providing a password so that the backup file can be encrypted. Use **adb restore backup.ab** to write the contents of the file back to your phone. Finally, to reboot your device into *Fastboot*, type **adb reboot bootloader**.

Fastboot

All the really exciting stuff happens with *Fastboot*, which is a command most commonly used to reflash Android's various partitions and to install different operating systems and bootloaders. And because it's always messing with the filesystem, *Fastboot* needs to be run outside of the operating system, which means restarting your device separate from both the normal boot and the rescue partition. Many of *Fastboot*'s functions will result in the loss of data, or even a complete rewrite/erase of your system, so it's important you're happy with these risks and have a current backup before going any further. Without the **adb** reboot, *Fastboot* is normally accessed by turning on your device while holding down specific buttons – power and volume up on Nexus 5, Ubuntu Phone, and you'll normally have to choose 'fastboot' from a primitive menu. You can then safely connect your device to your desktop with a USB cable.

With your phone/tablet in fastboot mode, type **fastboot devices**. Just like the output to *ADB*, you should see your phone listed as a device and you can now execute fastboot commands. *Fastboot* itself is a USB protocol and command language for performing tasks on your device without any operating system. One of the first commands you should try is **fastboot oem unlock**. Depending on your phone, this may erase everything and return your operating system to its default, but it also allows you to overwrite the recovery partition with something like the Clockwork Mod utility. These will be downloadable as **recovery.img** files from places like the Cyanogen Project, built specifically for your device, and they can be written to your flash memory by typing **fastboot flash recovery recovery.img**. This only replaces the **recovery** partition, and you can perform similar tricks with both the **boot** and **system** partitions, if you have access to properly formatted image files.

With older versions of Android, you could often unzip any updates and see the files that were going to be copied to the **system** partition. It was sometimes even possible to remove or replace those packages you didn't want installed into the read-only part of your phone. Recent updates for most phones will now compress these files into a **system.new.dat** file that no longer corresponds to the contents of the filesystem. Thanks to a great tool, however, you can still unfold this date, change the contents, and roll it back into a **system.img** file that can then be flashed using *Fastboot*. This tool is a Python script called **sdat2img** (**https://github.com/xpirt/sdat2img**), and you run it as follows:

```
sdat2img system.transfer.list system.new.dat system.img
```

The end result is a **system.img** file that you can send to your device with the **fastboot flash system system.img** command. As **system.img** is also just a block device formatted as ext4, you can also mount it to take a look at its contents. Before your device can boot, or at least access the Bluetooth and Wi-Fi drivers, you will also need to make sure the ramdisk and kernel for the system file flashed onto your phone/tablet. Luckily this is included in current **update.zip** files as a simple **boot.img** file that can be flashed by typing **fastboot flash boot boot.img**. It is possible to update devices using just the **system** and **boot** files, without going though the update process. This allows you to update to Lollipop on the MX4 Ubuntu Phone, for example, and it might allow you to create your own installations by removing those apps you don't require, as well as a possible installation route for Cyanogenmod for unsupported devices. But better than anything, it gives you the ability to explore the inner workings of the computer you keep closest.

FAQ

OpenStack

Ben Everard's favourite type of clouds provide snow. His second favourite run on OpenStack.

BEN EVERARD

Q OpenStack? That seems to be in the tech news quite a bit. What's it all about?

A OpenStack started as a project between Rackspace and NASA. The space agency wanted a private cloud computing system and needed the technology to manage it. That was back in 2010. Since then the then, the wider IT industry has adopted the technology and it's developed into a complete stack for cloud deployments.

Q When you say 'a complete stack for cloud deployment', what exactly do you mean by that?

A OpenStack isn't a single piece of software, but an ecosystem of components. These components each deliver a single aspect of cloud infrastructure. To list just a few:

- **Nova**, the compute component, enables you to launch and control virtual machines running on a variety of technologies (including KVM, Xen and Linux Containers).
- **Cinder** creates block-storage for use by the virtual machines created in Nova.
- **Trove** is the database component and enables you to quickly and easily share databases between applications.

At the time of writing, there were over 15 components that can be deployed, although any single OpenStack deployment doesn't have to use all of them.

As you can see, many of OpenStack's components are built on top of tried-and-tested data centre technologies. In most cases, they're also agnostic of the underlying technology, which means that you could use OpenStack to manage proprietary services running in a hosted data centre, or you could host everything yourself on servers you control. This puts the user in charge of technical decisions rather than locking them into a single technology.

Q How do all these components come together from a user's point of view? Is there some form of OpenStack interface?

A There's an OpenStack component called Horizon, which is a web dashboard that brings everything together into an easy-to-manage website. This enables you to manage your entire cloud deployment from your web browser. You don't have to use Horizon though. Each of the components has an HTTP API, so they can be controlled from other software. This API is compatible with Amazon's web services, so much of the software that's been designed to work with this commercial stack can also work with OpenStack.

> OpenStack is designed for enormous organisations (like NASA) that need to manage whole data centres or groups of data centres.

Q This all seems a little over the top. I can create virtual machines using *VirtualBox*, or Gnome Boxes. Block Storage is easy enough to manage, and I've never had a problem managing my databases. What does OpenStack give me that I can't do without it?

A The important thing to realise about OpenStack is the size of the situation that it's designed to deal with. If you have a single server running a few virtual machines and hosting all the storage on it, then there's no need for anything this heavyweight. OpenStack is designed for enormous organisations (like NASA) that need to manage whole data centres or whole groups of data centres.

When looking at it this way, it's not that OpenStack makes things possible, it's that it makes them easy to run and manage. There's nothing that you can do with OpenStack that you can't do something less buzzwordy, but rather than having to use different tools with different interfaces for each thing, you can use a standard tool for everything. If you need to write custom software to manage it all, you can use the standard API for this rather than having to deal with each item separately.

Q Let me get this straight: OpenStack is a bunch of pieces of software that I can use to set up my own private cloud services (virtual servers, backup storage, authentication, etc) on my own hardware?

A Sort of. It's the software to set up a cloud on servers. This could be a private cloud running on your own data centre, or it could be something you pay a hosting company to run for you. It could even be a public cloud that you rent out to a group of clients. The point of OpenStack isn't really that it's for public or private use, it's that it's an open cloud platform that you can do whatever you want with.

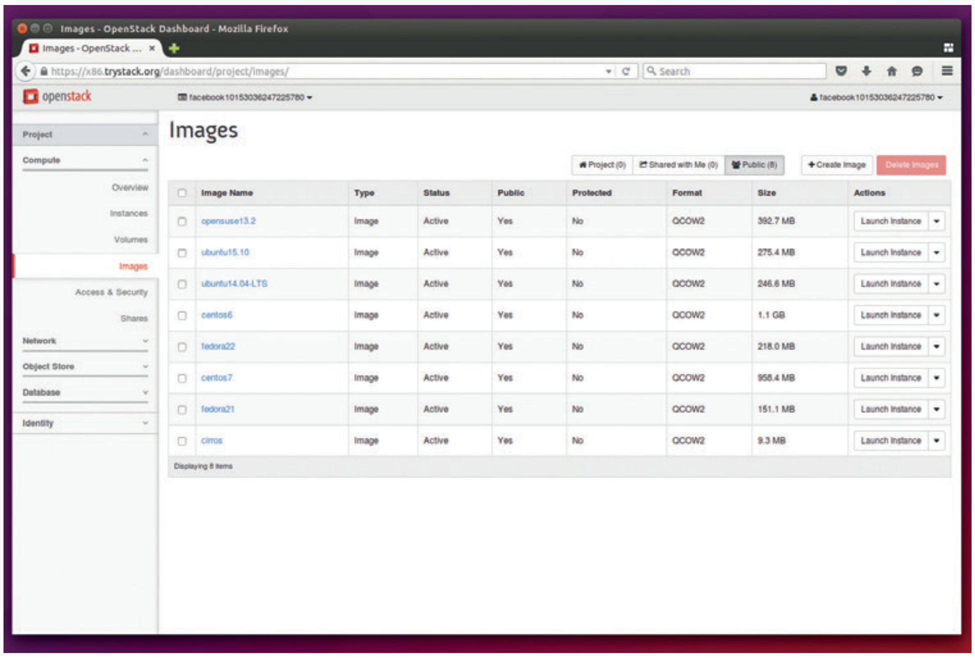
Q OpenStack sounds cool, but is there any way of trying it out without going to the not inconsiderable expense of setting up a whole data centre to run it?

A Yes! Although in production, you'd generally want several machines to run different parts of OpenStack (because if you don't have that level of infrastructure, you probably don't need OpenStack), there's a setup designed for getting set up and testing out on a single machine. This is called DevStack, and you can get it from **http://docs.openstack.org/developer/devstack**. There you'll find instructions on how to set everything up, but it's all quite straightforward. You just need to clone a *Git* repository and run a script.

Alternatively, there's a test system available at **http://trystack.openstack.org**. This enables you to spin up a few machines to see how everything works. At the moment, you have to log in via FaceBook to use TryStack, which make us love it a little bit less.

Q You said that NASA and Rackspace came together to make OpenStack, are they still developing it?

A The project is now managed by the OpenStack Foundation. Rackspace is still heavily involved in this, NASA less so. The foundation is run by directors and a technical committee, whose members come from a huge variety of companies. The Linux world is represented by people from Canonical, Red Hat and SUSE. From the wider technology world, IBM, HP and Cisco are all heavily involved as well. With so many of the big names in server computing involved, you can be sure that OpenStack will be around for quite some time to come.


Images - OpenStack Dashboard - Mozilla Firefox

https://x86.trystack.org/dashboard/project/images/

Images

	Image Name	Type	Status	Public	Protected	Format	Size	Actions
☐	opensuse13.2	Image	Active	Yes	No	QCOW2	392.7 MB	Launch Instance
☐	ubuntu15.10	Image	Active	Yes	No	QCOW2	275.4 MB	Launch Instance
☐	ubuntu14.04-LTS	Image	Active	Yes	No	QCOW2	246.6 MB	Launch Instance
☐	centos6	Image	Active	Yes	No	QCOW2	1.1 GB	Launch Instance
☐	fedora22	Image	Active	Yes	No	QCOW2	218.0 MB	Launch Instance
☐	centos7	Image	Active	Yes	No	QCOW2	958.4 MB	Launch Instance
☐	fedora21	Image	Active	Yes	No	QCOW2	151.1 MB	Launch Instance
☐	cirros	Image	Active	Yes	No	QCOW2	9.3 MB	Launch Instance

Displaying 8 items

OpenStack's Glance module enables you to store images that you can use to boot virtual machines at the click of a button. Think of yourself as the Ernst Stavro Blofeld of cloud infrastructure deployment.

Q With all those competing companies on the board, has development ground to a halt as bureaucracy and politics engulf everything?

A Not at all! There's a new release of OpenStack every six months. As we write this, Kilo is the latest release, but Liberty should be out by the time the mag hits the shelves (the project follows an alphabetic naming process). Even with the six monthly releases, each new release packs quite a lot of features. To give you an idea, the Kilo release notes were over 6,000 words of dense feature descriptions. That's an impressive amount of new stuff for just six months' work.

The rapid advancement is no doubt due to the young nature of the project. Five years may seem like a long time, but it's just the blink of an eye in enterprise software development. Over time, we would expect the number of new features in each release to slowly tail off as the software becomes more mature.

Q This all sounds very impressive. Are there any competitors, or does OpenStack stand alone?

A The most direct competitor is Apache's CloudStack. This is another cluster of open source components designed to enable people to roll their own cloud environments. Cloudstack, however, is significantly less mature than OpenStack and has far fewer contributors and deployments.

The most widely used competitors to OpenStack are all closed source, hosted cloud infrastructures such as Amazon's Web Services (AWS), Google's Cloud Platform and the Microsoft Cloud. None of these are direct competitors though, because they all involve letting another company host your cloud for you, in contrast to the control that CloudStack gives you. Using one of these options means potentially locking yourself into a system that's hard to leave and losing control of how your computing resources are hosted.

Q So, in a way, OpenStack is kind of like the GNU/Linux of the enterprise computing world? It's a set of open source components that let you take control of your information technology rather than surrendering it all to a commercial entity.

A Exactly!

“The more effectively you participate in Free Software, the more benefit the project will get out of it, and the more benefit you’ll get out of it.”

ALLISON RANDAL
PRESIDENT OF THE OSI

We meet the new president of the OSI, Ex-president of the Perl Foundation, chief architect of Perl's Parrot and long-time program chair for OSCON.

The open source conference, OSCON, started out as the Perl Conference, which made Allison Randal's tenure as its program chair particularly fitting. She'd worked with Larry Wall, the creator of Perl, for a long time, and has been instrumental in Perl's success, partly thanks to her work on the Parrot virtual machine. But Perl is just one aspect to her career in open source. As well as working at O'Reilly, editing and publishing her own books, she's worked for Canonical and currently works at HP. Allison has also just taken over from our own Simon Phipps as the president of the Open Source Institute, which meant that when we discovered we were in the same time zone has her, we had to grab a few moments of her time.

LV Do you miss being the program chair for OSCON?
Allison Randal: I do, I do! It's like a little nostalgia when I come here every year. It's like the old home camp.

LV It must have been difficult giving that up, and for every other outgoing chair.
AR: We have a tradition of skipping a year. Nat (Torkington) skipped a year when he created it, I skipped a year when I created it, I just didn't even attend. But they're doing a great job. Now it's actually quite exciting to come back and see it. They're carrying on with it and still going strong.

LV You've done quite a few things in quite a few places, and it's fascinating that you started off as a linguist – this seems to be a Perl thing, what with Larry Wall doing something similar. What is it about Perl that attracts linguists?
AR: In my case it was a very direct connection, which is where I met Larry and we started talking about linguistics, and that was why I got involved in Perl. Although, I guess I was using it at a startup before that, but that was why I got involved in developing Perl. So in that case, yes, it was absolutely because I was a linguist that I got pulled into it. I think the way Larry thinks, he thinks about language and he thinks flow of language, so the way you wrote constructs for Perl was very like a natural language, whereas a lot of computer languages don't really have that in mind – instead, Iit's just like, "how do I mathematically express this problem". So I do think it does appeal to, not just linguists, but poets and writers that make that transition.

LV It's not the easiest programming language though, so we're surprised that any non-technical people would find some kind of fluency in Perl.
AR: Think about how complicated English is! I think, if you don't keep it very clean and simple, it can be complicated and you can end up making this very complicated program that no one can ever understand, and that's why Perl kind of ended up getting a bad reputation. You can write a very dense legal document that's almost impossible to read and it's the same language as a children's book, but you've just expressed yourself in a way for a certain circumstance.

LV Do you think the same will be true for Perl 6 and its multi-paradigm capability?
AR: I haven't been following it closely lately, but when I was involved in the design of Perl v6, we did have a goal of making it easy to understand. That whole getting started easily was still very much a very big part of it, and I feel like it still had that. It went through a cycle of getting very complicated and it seems like it kind of came back around to being more focused on being easy to use. I don't think it's so much of a

OSCON, the gathering of geeks on America's West Coast, wouldn't exist without Allison's guiding hand, which means we wouldn't have developed such a taste for US-style IPAs.

As the former Technical Architect of Ubuntu, Allison's done more than most to make Linux and Free Software approachable by non-techies.

problem having those paradigms mixed in, in much the same way that the early Perl mixed in a lot of paradigms. It mixed in like procedural and then it added object orientation onto that.

LV Do you still do much Perl development?
AR: Not much, no. My last job before I moved to HP, I was a CTO at a startup that was all Perl.

LV So that was after Canonical?
AR: Yeah, it was in between the two. But at HP I haven't done a whole lot of development. My main role is open source strategy. One of the big things I do is that I maintain this 30-page strategy document that covers all the open source projects that HP invests in.

It starts by setting the principles, like contribute upstream first, and why we're doing what we're doing, but then it also goes through each project and lays out what the project is, here's why we care about it, here's our priorities, here's where we're investing and why, sort of keeping open source as a fundamental part of the business. It's a place where you invest and get value. Just like anything else, you need to have that same consciousness that you have with monetary resources or human resources.

LV HP was one of the biggest users of Ubuntu and one of the few to actually pay for it. In what other ways does HP use open source?
AR: I don't cover the whole company, I cover HP Cloud. HP's cloud product is OpenStack and it's almost pure upstream OpenStack. There's a few patches, but they consciously push them back upstream as fast as possible. And there's also Cloud Foundry, which is another big one that we've heavily involved in.

Recently we've switched over to doing deployment with Ansible (orchestrated configuration management for networks), that's another big one that we've been very involved in. Debian too, for the machine stuff like trying to explore new kinds of hardware for storage, where you can merge memory on disk, so you get fast disk and flexible allocation. So they're doing a lot of work on the Linux kernel and on Debian to try to have a fully functional distro on this hardware that's still very experimental.

LV What would you consider the biggest successes and failures of open source?
AR: The biggest success is that we did actually get corporate adoption. That was the beginning of open source. We've really struggled since then, since that was about 2010, to find what we want to do next. It turns out there hasn't been an obstacle. There's a survey by Black Duck [**www.blackducksoftware.com**] that shows corporate use. So 2010, it was 48% of respondents said they'd use open source, in 2015 it's 78% with about 68% contributing and 88% saying they plan to use or contribute to it in future. So it happened anyway.

Companies are driven to contribute upstream for practical reasons

LV For the current generation of developers it's the only way to work. That's amazing. We never thought that would happen. But it brings bigger challenges. We've heard copyleft licences being talked about by open source developers as being viral, almost for the first time in years (this was at a Lightning Talk one evening at OSCON), which is worrying. And maybe there's a move towards permissive licences because they feel that the job has already been done.
AR: I'm kind of in two minds on that. Companies are driven to contribute their changes back upstream for

practical reasons… for example, forks are very expensive to maintain, so pushing your patches upstream reduces the cost of maintenance, and fixing bugs upstream and adding features upstream benefits the company. So there's that economic drive to contribute your changes back anyway. To a certain extent, I think copyleft in the 80s, when it was introduced, was essential.

We wouldn't have got here without it.

AR: Not all the time, but in many cases you can get the same effect without it, so I'm not worried about permissive licences. On the flip side, I think the fact that companies have learned that they have to contribute their changes back anyway, for practical reasons, will also make it them less afraid of copyleft because it's no longer a scary thing to have to contribute your changes back, it's just normal and what they have to do anyway.

Do you think the initial momentum for the GPL to make sure everything is contributed back might be lost?

AR: GitHub recently did a licensing survey and they found that the MIT Licence was the most common licence on there. I think the important thing to do is keep in mind that it is very important to educate these consumers of open source and free software that they need to contribute their changes back [to the projects].

Where there's disagreement, it's part of Allison's job at the OSI to remind us that we're all on the same side.

Is that something that you want to do as part of your role at HP and as president of the Open Source Initiative?

AR: Yeah, it is. That's kind of my whole role at HP, is like, "No, no. Look it's in your best interests to contribute your changes back,". Whatever the licence, it doesn't matter, you still need to contribute back.

And is HP happy about pushing changes back up stream?

AR: Yeah.

Is that something that's changed recently?

AR: It's an attitude that's changing all over, it's not specifically HP. But since about 2000, yeah, it's an attitude that has changed. It's just coming to grips with the nature of the software industry. In some sense, they see their competitors doing it, and they're seeing all this money, and then we're going to be left out if we don't follow, and then it ends up spreading and spreading.

So where is the challenge if most companies get it?

AR: The challenge is that they all get it but they don't all get it to the same degree. There's a difference between understanding software freedom and just being driven to it by economic necessity. That's part of why I talk about education. You still need to learn the full benefit cycle, and you still need to learn how to participate effectively because the more effectively you participate, then more benefits the project will get out of it, the more benefits you'll get out of it.

The way I explain it that seems to appeal to people is that it's a competitive edge. So using Free Software and open source used to be the competitive edge, but now everybody does that. So now the competitive edge is participating effectively, and if you're participating more effectively than your competitors, you will get more bugs fixed, you will get more features added, you will get your problems solved, you will have an edge over your competitors and it will benefit business.

Is there a good example of a company that gains that

You can watch Allison's keynote from OSCON here: www.youtube.com/watch?v=egEO1L8EHJU.

competitive edge? For example, we like Canonical but it's often criticised for not doing enough, certainly upstream.
AR: If you look at the history of their company and their business, a lot of the things they struggled with have been specifically in the areas where, if they had been contributing regularly and had been participating in the wider world, they could have just eliminated that entirely. They never would have had a problem in the first place.

LV But they do get open source.
AR: They do, and that's what's so hard. They're one of the companies that totally gets software freedom, but they're not participating effectively and it's hurting them, which is sad. They're doing fantastically.

LV Has there been any failure that we could learn from?
AR: To my mind, the biggest failure has been the tension between Free Software and open source. I think that has actually hampered both movements in a way. The fact we have different emphasis is fine, I don't think that's a problem. But for a long time, and not so much any more, it's less and less over time, but for a long time the two movements kind of undercut each other, which was too bad. And I think it slowed the overall progress of things.

LV We think you're right, but it's still a difficult and thorny subject. We greatly respect Richard Stallman and, in an ideal world, he's right. But if he's not the best ambassador, we perhaps need his unmovable principles.
AR: Yeah, in some ways it's like John the Baptist. Not everyone's where he is, and probably we'll never be where he is, but just the fact that he's there and sticks to his principles, shifts the whole system over in that direction.

LV Free Software needed the formality of the rules system that Richard Stallman created to be able to create a framework, which is probably becoming relaxed now.
AR: Yeah, that's an interesting thought. I've been thinking a lot about inevitability of various things lately, about specifically the open source and Free Software movement and what was inevitable. And maybe it was inevitable that some kind of split would happen. One grew under another name or some kind of schism happened, but I still hope that we merge back over time.

LV Is there anything that you can do as president of the OSI to be able to build that bridge?
AR: We work closely with the FSF (Free Software Foundation) and we do collaborate on things when we can, like amicus briefs for various cases that have significance for both movements.

> Canonical is one of the companies that totally gets software freedom

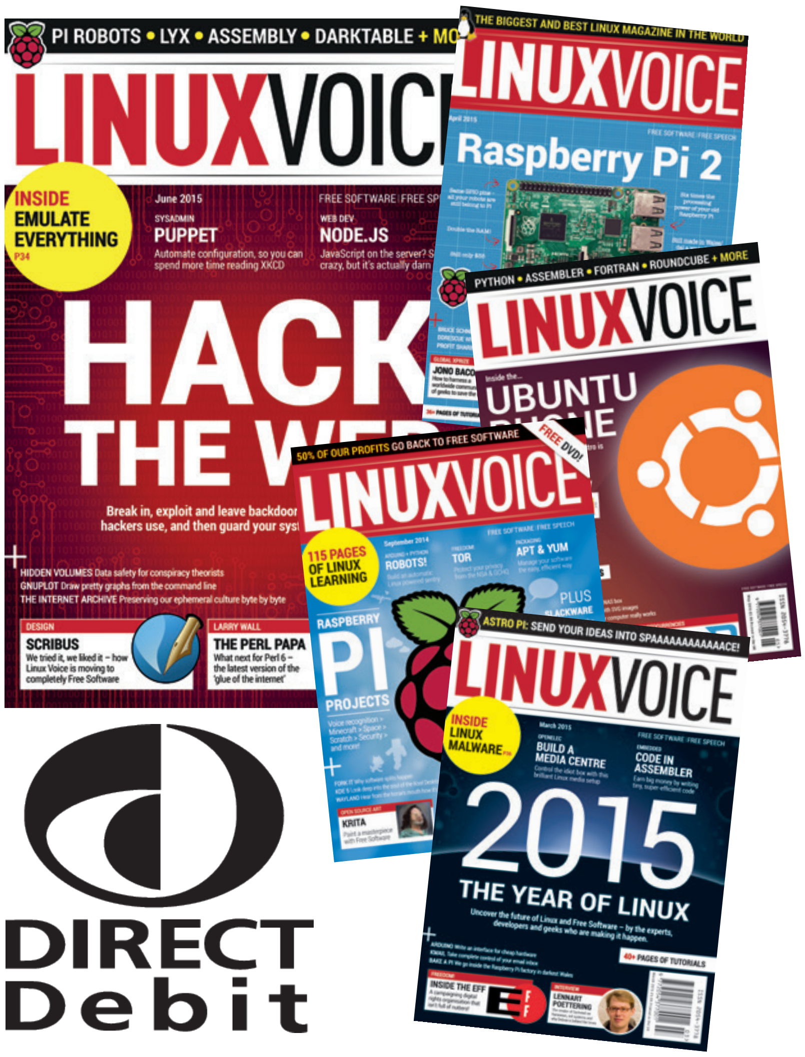
The blog I'm writing, connected to my keynote, explicitly talks a little bit about the history of Free Software and open source and the drivers for the split, but it also ends on the note that, as free software and open source succeed then we don't really need to talk about tactics so much anymore. We all agree that we want software freedom to win and that there are inherent problems with proprietary software, but the free software movement is focused on boycotting and a specific set of tactics.

LV Do you think the FSF is becoming more independent and using a more pragmatic approach before it becomes too marginalised?
AR: I've been pretty pleased over the past few years to see what's coming out of the FSF. I'm good friends with John Sullivan, he's the executive director (see Linux Voice issue 19 for our interview with John). He's a moderating influence. He really does represent the future of Free Software.

Subscribe

UK READERS!

Did you know that you can subscribe to Linux Voice from just £10 per quarter with Direct Debit? Get every issue straight to your mailbox (or inbox) and spread the costs!



DIRECT Debit

What you get

- 100 pages each month of the best tutorials, features and interviews
- Access to all back issues in DRM-free digital formats - over 1,500 pages
- Take part in our yearly profit donating scheme, and help FOSS projects

Yearly Direct Debit prices

UK print subscription – £55
Digital subscription – £38

Quarterly Direct Debit prices

UK print subscription – £15
Digital subscription – £10

Go here now to subscribe!

www.linuxvoice.com/shop

Payment is in Pounds Sterling. If you are dissatisfied in any way you can cancel your subscription at any time and receive a refund for all unmailed issues.

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
Studied economics and psychology at one of the UK's most prestigious universities.

You don't need to have studied economics and psychology at one the UK's most prestigious universities to see that the egregious fraud perpetrated by Volkswagen is unlikely to be a one-off event. Any car maker doing the right thing, working hard and playing by the rules is going to be at a disadvantage to the companies that cheat, so the incentive to cheat is enormous. I would be hugely surprised if more auto makers weren't involved in this disgusting scandal, and if I were a Saab shareholder I'd be even more annoyed with VW than I am now.

Of course, they were only able to get away with it for so long because the software is closed source. If we were allowed to look at the source code for the software in our cars, the increased transparency would put pressure on manufacturers to be as efficient as possible, knowing that any false claims could be scrutinised.

Of course, that's not the way they see it. But if your reputation is built on reliability, it's an awful shame to throw it away for the sake of an outdated software development model.
andrew@linuxvoice.com

On test this issue . . .

42 Bitwig Studio 1.2

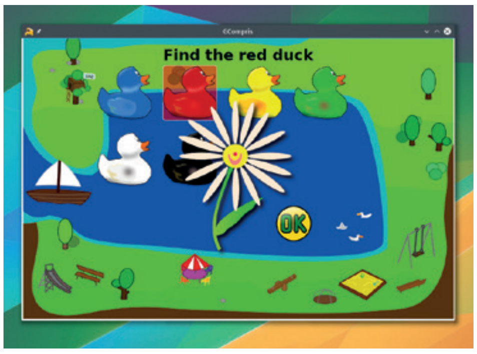
Ableton Live gets a worthy competitor, and, amazingly, one that runs natively on Linux. We can't stress enough how pleased we are that this excellent audio production software is here. What a time to be alive – praise be to Saint Moroder!

Gnome 3.18 43
More features, more refinement, more silly characters – the little desktop has all grown up.

Unity 44
If you feel like making squillions of pounds coding the next AAA game, why not use this engine?

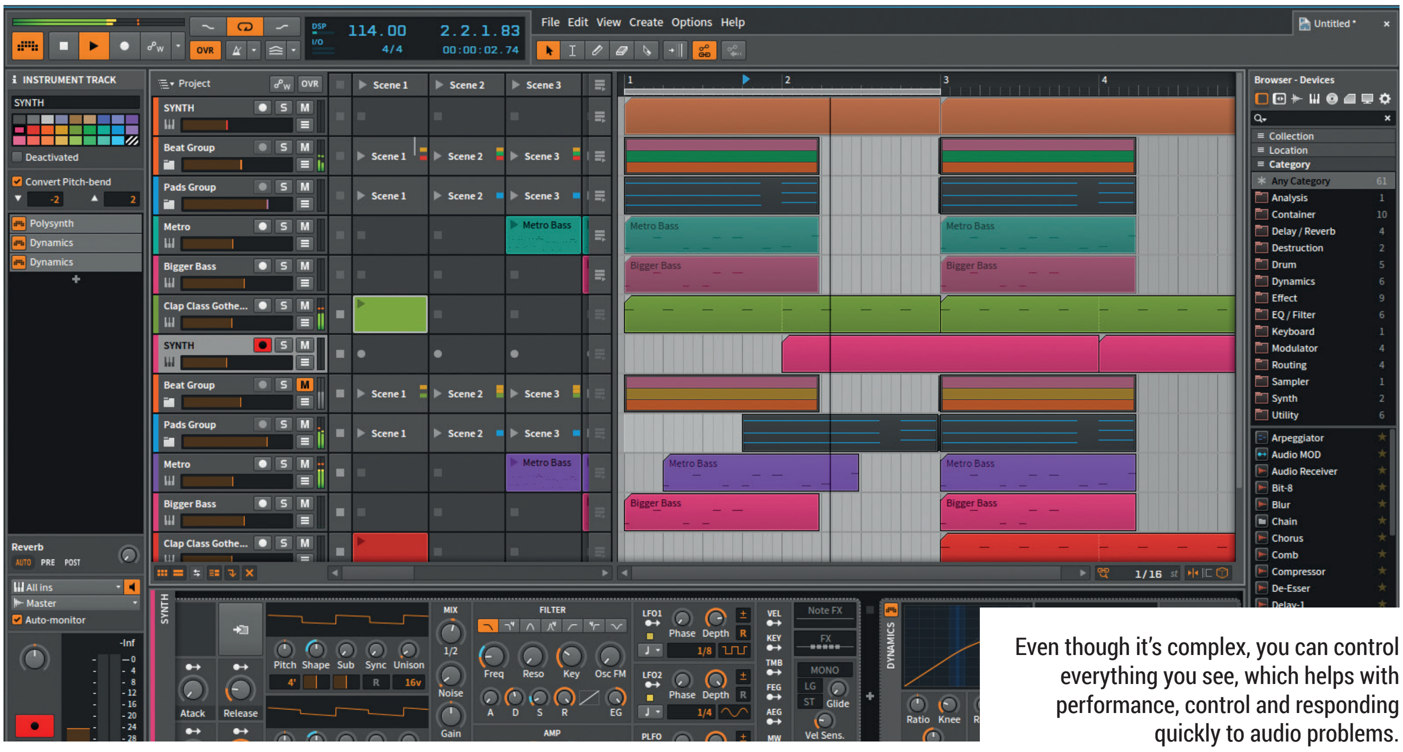
Devolo 1200+ 45
Powerline Ethernet has never been so reliable, so easy to use, so effortlessly jazzy.

Group test and books



Group test – educational software 50
Computers for kids aren't just idiot-proof iPad distractions – there's some serious learning potential in even the lowliest Linux machine.

Books 48
Praise be, for *Hello Ruby* has arrived! Only a year and three quarters since its author crowdfunded $380,000 to publish it. We hope it's good.



Even though it's complex, you can control everything you see, which helps with performance, control and responding quickly to audio problems.

Bitwig Studio 1.2

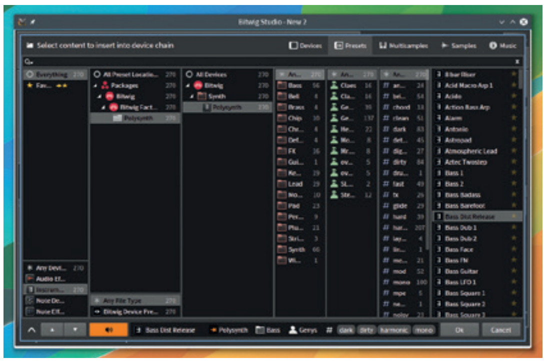
Still in rapture, **Graham Morrison** tests an update to the best audio software around.

Web www.bitwig.com
Developer Bitwig GmbH
Price €259.00

Version *1.0 of Bitwig Studio*, which we reviewed in issue 4, was a huge milestone for the audio production landscape. Eighteen months later, there have been several significant updates, but none have been on the scale of this 1.2 release. It's a major overhaul, adding all kinds of improvements to what was already a unique implementation of a music production environment.

What *Bitwig* does (if you don't have your CC-BY SA copy of issue 4 handy) is build pieces of music out of audio and MIDI clips, internal and external sound sources, effects and virtual instruments, allowing them to trigger and arrange them into both scenes and more traditional timelines. And it's brilliant.

The first impressive new feature is that, like *Ardour*, the *Jack* audio connection kit is no-longer a prerequisite. You can now start *Bitwig* normally using your audio hardware's ALSA drivers, which is especially useful for laptop music making, where the on-board audio rarely works well with *Jack*. Our second favourite change deals with the practicalities of audio production, and that's the ability to group tracks together. On medium-sized projects, and even our podcast, putting tracks together in what is effectively a virtual folder is an essential. This is where we keep the various parts of Brad Sucks' awesome music, for instance, so that his tracks don't get in the way of the other audio. *Bitwig*'s implementation is a little clunky, especially compared against *Cubase*, but it's a step in the right direction.



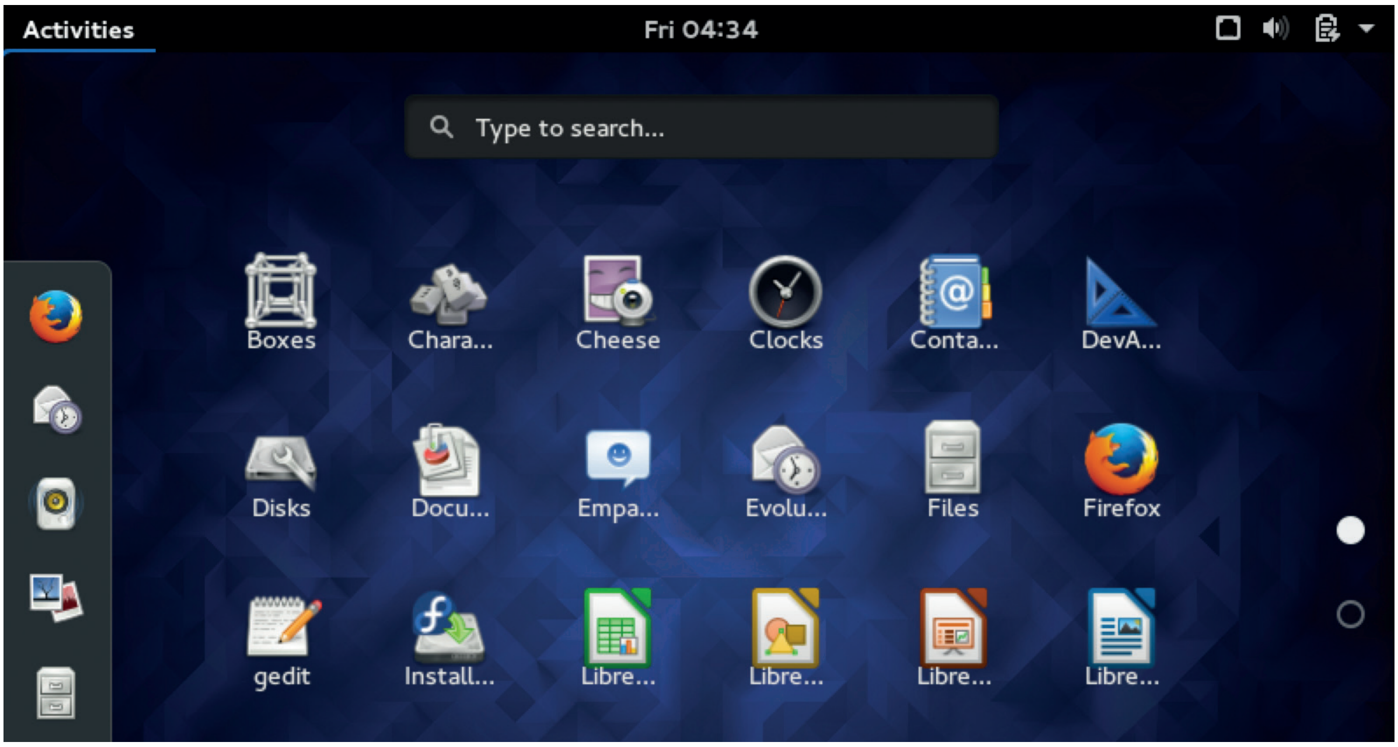
The preset browser now gives you an instant preview of audio and effects, helping you choose the right sound for each new track.

Love to love you baby

We also love the way you can now audition audio, effects and sound generators from the new browser. It works perfectly and is the best implementation of this idea we've seen. The new oscilloscope plugin is also brilliant for troubleshooting, it lets you to visualise audio even when you can't hear it. The huge range of new audio packs, the new Delay 4 effect, and presets give you more options than ever for creating music. *Bitwig* is expensive, but if music production is your thing, there's nothing like it on Linux.

Turn your Linux laptop into an audio production, nightclub DJ, synth and effects workstation.

★★★★★



Gnome 3.18

Like clockwork, a new Gnome release is here. **Mike Saunders** checks it out.

Web www.gnome.org
Developer The Gnome Project
Licence GPL and LGPL

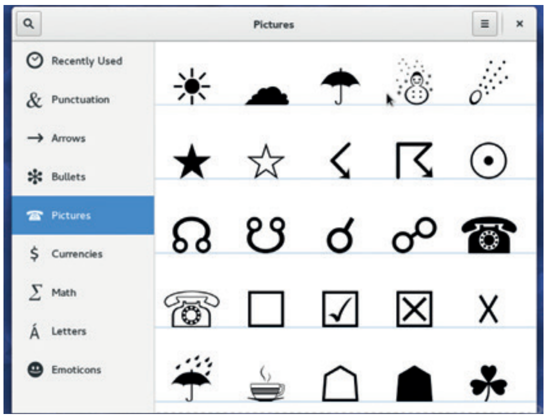
These are good times for Gnome. Early releases in the 3.x series caused much consternation, but after years of improvements, we know many Linux users who abandoned Gnome early in the 3.x cycle but have since returned and are loving it.

Gnome 3.18 contains a mixture of new features and refinements. The file manager now sports Google Drive integration and has seen lots of work to improve usability, with progress windows for file copy/move operations replaced by a single button in the toolbar, and an "Other Locations" entry in the sidebar for commonly used network locations. File and folder renaming and creation is now smoother, using dialog boxes and popovers.

Two new programs have been added to this release: Calendar does as its name suggests, with support for linking to online calendars and switching between month and year views. Then there's Characters, a browser for numbers, letters and Unicode symbols. If you like to paste pictures of pandas into your emails or IRC chats, your life will become slightly easier with this release.

Hardware-wise, Gnome 3.18 has a couple of new features worth shouting about: automatic screen brightness is now supported, so if you're on a laptop with a light sensor, Gnome will adjust the brightness to match your surroundings (and potentially save battery life). It's also now possible to update your machine's firmware (eg your BIOS) from a GUI thanks to the Linux Vendor Firmware Service initiative, drastically simplifying a job that normally requires much fiddling at the command line.

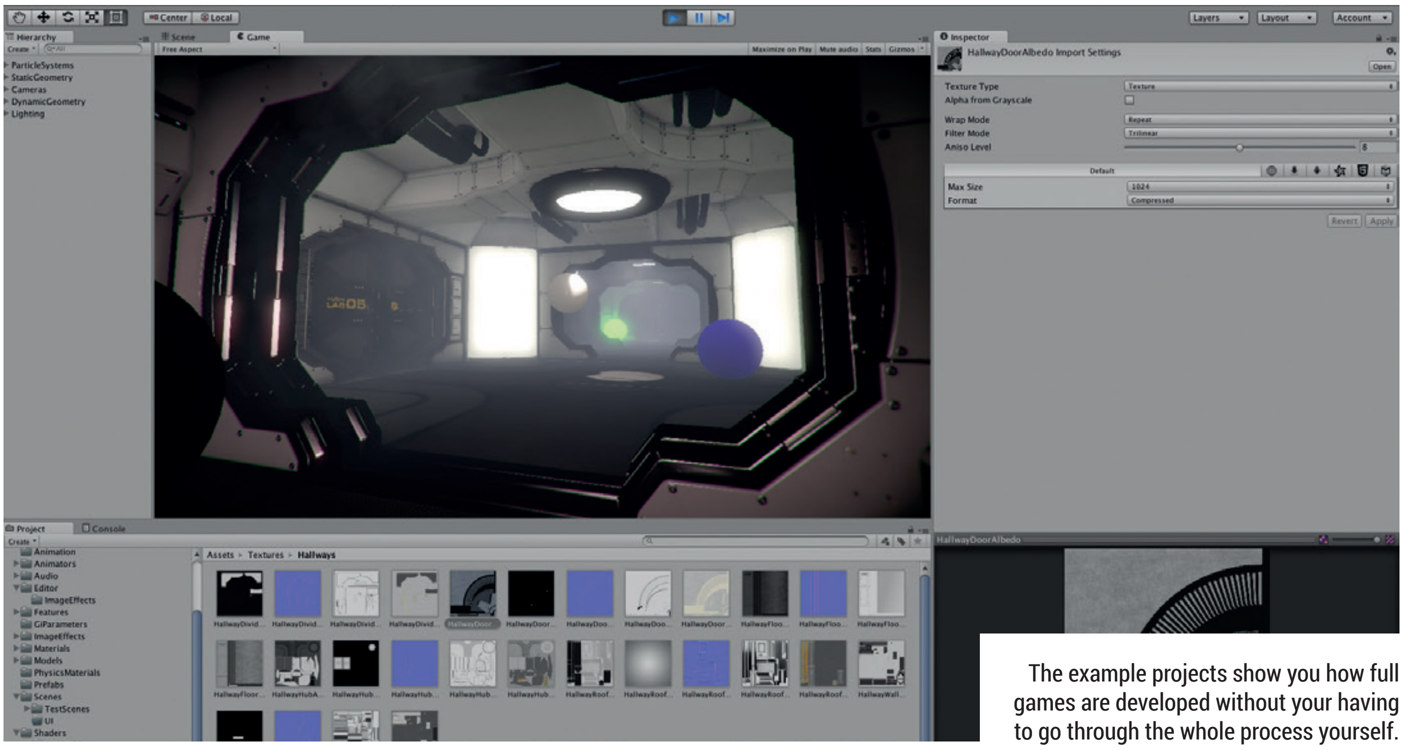
Gnome 3.18 is a solid update from the Gnome team, and we'll give them extra credit for their communication efforts. The Gnomers really work hard to demonstrate what's new in each release, with short videos and attractive release notes. If you're one of the ex-Gnome users who gave up on the desktop a couple of years ago, now's the time to give it another shot.



Characters is one of the new additions to this release: a trivial little tool for finding Unicode characters like, er, snowmen.

Mostly refinements in this release, with a couple of new goodies tossed in to round it off.

★★★★☆



The example projects show you how full games are developed without your having to go through the whole process yourself.

Unity 5

Now that a new games engine runs on Linux, **Ben Everard** gets interested in coding.

Developer Unity Technologies
Web www.unity3d.com
Price Free or $75 per month

Unity is an Integrated Development Environment (IDE) for developing games, which comes complete with an engine to handle most of the tricky parts of games development (such as animations and physics). When combined with *MonoDevelop* (which handles the C# scripting), you get everything you need to make top-tier games.

Unity is free (as in zero cost), but only if you or your company earns less than $100,000 per year. Once you pass this threshold, you need to pay $75 per month plus an additional $75 per month for each mobile platform you want to support. Alternatively, you can pay $1,500 (plus an additional $1,500 per mobile platform) to buy a licence outright.

Games development in Unity is mostly point and click to bring various assets and animations together in the game engine that can handle collisions, interactions, gravity and most of the aspects of creating a world in which the game exists. C# scripts can then be used to fine-tune this world and the gameplay within it. As a professional-standard games engine, *Unity* gives the user a huge amount of control over how everything is done, and this leads to the interface being quite complex. Fortunately, there's good documentation, and a good selection of tutorials to help new users get started.

Unity includes an asset store that enables users to buy and sell parts of games including 3D models, animations and particle systems, which can help you to very rapidly build up a high-quality game even if you have little artistic flair. However, it does add another layer of proprietary licences to your game.

Learning *Unity* is a serious task, and someone interesting in playing around with games development would probably find it much easier to use something simpler, perhaps even just a games library in a language they already program in such as Pygame. Freedom-loving developers will also be put off by the licence. However, if you're interested in serious games development, and are happy relying on a proprietary engine, *Unity* is probably the best option on Linux at the moment.

The asset store is bundled with Unity or viewable in an external browser at **www.assetstore.unity3d.com**.

Linux gains a great development environment, but one that comes at the expense of freedom.

★★★★☆

The size of these devices is a little prohibitive, and we'd always like more Ethernet ports, but the extra power socket has proven very useful in tight installations.

Devolo 1200+ Wi-Fi Starter Kit

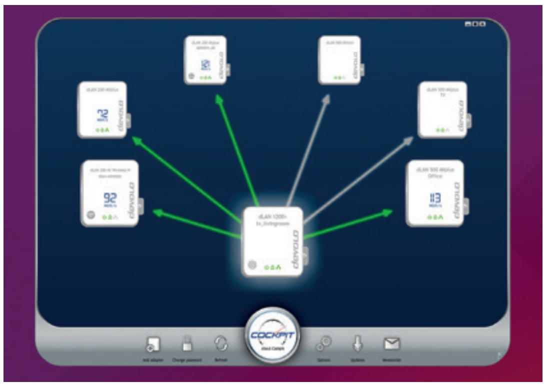
Graham Morrison finds a way to work from the shed at the bottom of the garden.

Broadband adoption is at critical mass and everyone wants Netflix in high definition delivered to their bedrooms. But Wi-Fi has problems with both range and bandwidth, especially in the average household with lots of devices, unfriendly partitions and microwave dinners. Which is why powerline Ethernet adaptors have become such an essential part of home networking. They pipe your network through your electricity cables, turning any power sockets into RJ45 Ethernet ports. Or in the case of the this package from Devolo, double Ethernet and a Wi-Fi access point.

The kit we've been sent includes two adaptors – one intended to work as a hub and containing a single Ethernet port, and another with two Ethernet ports and a comprehensive Wi-Fi access point. It's perfect for extending your network more than a single floor upwards, or into the garden. In fact, this was how we tested the kit, and the 1200+ Wi-Fi adaptor proved faultless. You press the button on a device that's already part of the network, run down the garden, quickly plug in the new adaptor and press its own button, synchronising with the existing network.

Devolo in the detail

There was a very real and repeatable improvement over the pre-500av era of Devolo devices, and it was a similar story with speed. Download speeds in the shed were around 150Mbps, increasing to 250Mbps with a closer socket. These figures are way off the 1200 hinted at in the product name, but we're immune to this arbitrary rating and these units are much better than other devices we've tested. 2.4 and 5GHz concurrent Wi-Fi was equally simple through the web interface and we were very impressed by the access point's capabilities. A physical button toggles the Wi-Fi, and there's a scheduled timer too, plus guest accounts and excellent parental control. This kit is expensive, but we feel that in this bleached white plastic case, you get what you pay for.



Despite using the deprecated Adobe Air, Devolo's Cockpit utility does provide a useful overview of speeds between devices and the Ubuntu installation is easy.

Web www.devolo.com
Developer devolo AG
Price £160

If you need network connectivity in a remote part of your estate, these units perform most excellently, and are worth the expense.

★★★★☆

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

PLAYING CATCH-UP

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

The 1,500 Linux games on Steam milestone has been reached, and though this doesn't seem much compared with the 6,500 or so games for Windows, it is indeed very significant. At the same time, this can only grow over the coming year, with SteamOS on the horizon and the coming improvements on the AMD driver front, as well as a greater commitment to the Vulkan API, with the latter being increasingly discussed publicly by the big hardware companies.

What the numbers don't show, however, is a fundamental change in the types of games we're seeing on Linux. A year ago, the Linux gaming space was dominated by indie titles, but there are now a great deal of large developers porting in-house or outsourcing to a number of porting houses, most of which also port games to OS X. Along with this, spending habits are also changing, with Linux gamers buying fewer games at higher prices as more people move from bundle-style games to AAA mainstream titles, which are considerably pricier.

There are still some very big-name developers missing though, such as Blizzard, Bethesda and Rockstar. The biggest is certainly EA, which joined Blizzard recently in stating that it has no plans to support Linux until the market share is bigger – though the acknowledgement suggests that many out there are keeping an eye on Linux Gaming to see how it's evolving, and waiting to see if SteamOS and Steam Machines are a hit.

Company of Heroes 2

A solid World War II RTS game.

There aren't a whole lot of Real Time Strategy games on Linux, especially not of the calibre of a franchise like *Company of Heroes*. The game focuses primarily on the eastern front playing as the Soviets, with a few smaller single-player and multiplayer campaigns taking place on the western front and more available as downloadable content.

Focussing on the eastern front is a refreshing choice, given that certain parts of the war have been done to death in WW2 games. The story the game presents, where an imprisoned Soviet lieutenant recounts his experiences, isn't particularly profound, but does well in outlining the events of the war to those not entirely familiar with it.

The gameplay in *Company of Heroes 2* is excellent, and shows that RTS is the best way of conveying warfare and all its chaos, with a persistent pressure on the player as fascist troops storm the city or as soldiers desperately scorch earth before enemy troops advance. Similarly, the sheer expendability of the Soviet troops portrays the harsh and desperate realities of the front.

All the major fronts in the European theatre of war are playable – this looks like North Yorkshire.

The game focuses controlling small squads rather than entire armies, however it is also not uncommon for numerous simultaneous skirmishes to occur, testing the player's multitasking abilities under pressure. There are also some great features which reflect the harsh realities of the front, such as the extreme cold slowing troops down and eventually killing them, forcing the player to find sources of warmth on the battlefield.

Website http://store.steampowered.com/app/231430/ **Price** £29.99

The fast-paced action often necessitates a lot of multitasking.

Real-time strategy (RTS) is the best type of game to convey warfare and all its chaos

Satellite Reign

A worthy spiritual successor to the Syndicate series

A lot of game franchises have been brought back from the dead recently, and *Syndicate* is the latest to undergo the "spiritual successor" treatment. One of the things *Satellite Reign* does best is portraying its dystopian cyberpunk world ruled by mega-corporations with private police forces.

Unfortunately, its superficial plot and lack of characters makes many of its positive points feel like lost opportunities. Though there is a story, the game is more of an open world where players attempt break-ins into corporate facilities.

Unforeseen situations cause plans to go out of the window as the player has to quickly adapt to circumstances. There is also a good sense of progression as stats and weapons are upgraded and new skills learned, but after a while, many of the missions start to feel repetitive – something which could have easily been fixed with more dialogue and story. However, *Satellite Reign* is a great game for those who enjoy squad-based tactics.

Website http://store.steampowered.com/app/268870/ **Price** £22.99

Satellite Reign has one of the most convincing cyberpunk worlds seen in gaming.

The Stanley Parable

Dark, puzzling and hilarious.

In this game about choice and free will, the player takes control of Stanley, employee #427 of an newly abandoned office whose employees formerly followed orders from a screen, mindlessly pressing buttons on terminals.

The game is short and simple, and being too descriptive would give a lot away, but in essence the story is told through the choices the player makes. There are numerous possible endings, and despite some taking only a matter of minutes to find, it can take a few hours to find them all. As clichéd as this has become in recent years, *The Stanley Parable* is more of an experience than a game, and though it was one of the first in this new genre, it's still one of the best.

Other than the game's portrayal of choice in video games, its major strength is its dry humour, with subtle breakings of the fourth wall provided by the narrator – the only character in the game with spoken dialogue and the closest thing it has to an antagonist. Combining the humour with the experience is why this game is so highly recommended – it's thought provoking and fun.

The story's many paths and endings all begin at desk 427.

Website http://store.steampowered.com/app/221910 **Price** £9.99

ALSO RELEASED…

Adventurezator: When Pigs Fly
This "what you see is what you get" point-and-click adventure game maker enables just about anyone to make their own game with no programming skills whatsoever. At the same time there is also a (pretty good) adventure game included and the ability to play other people's creations on the Steam Workshop. There are a tonne of nice features that give the ability to create professional-looking games.
http://store.steampowered.com/app/300280

Train Valley
On the surface, this game looks like a railroad sim, like *Railroad Tycoon*, but it is actually a micromanagement puzzle game. It's a nice little casual game that's visually appealing and tonnes of fun for those not willing to commit tens of hours into a game. It still shouldn't disappoint fans of railroad simulators – if one expects a different style of play – given that there are many fundamental similarities.
http://store.steampowered.com/app/353640

Dropsy
A bit creepy, a bit strange, but also oddly heartwarming, *Dropsy* is the tale of a clown who wants to help people and give hugs. The game employs traditional point-and-click gameplay and graphics, though the story is far from traditional, with many underlying themes, some of which are pretty dark. The game does a lot of interesting things and it's worth getting if you enjoy creativity in video games.
http://store.steampowered.com/app/274350

A Peek At Computer Electronics

Ben Everard learns not to poke around inside a desktop case using his finger.

Author Caleb Tennis
Publisher Pragmatic Bookshelf
Price $22
ISBN 978-0-9776-1668-8

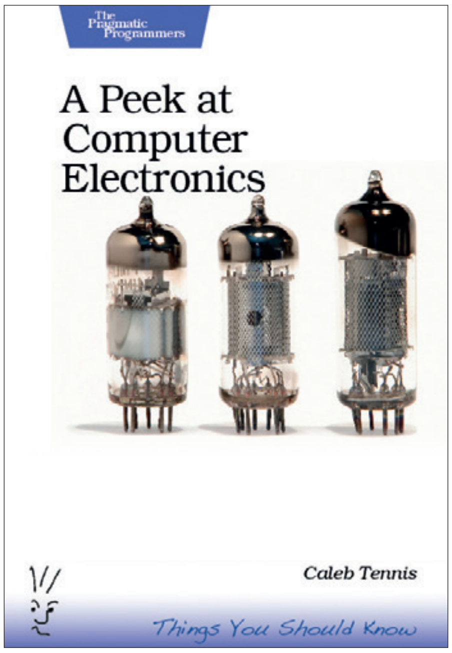
Author Caleb Tennis takes the reader on a 250-page journey from the discovery of electronics to semiconductors and the inside of a modern computer case. That's a long journey and 250 pages is not a lot to accomplish it in. There's a clue in the name here. *A Peek At Computer Electronics* really does nothing more than give the reader the briefest of overviews of this enormous subject.

By being rushed through everything, the reader is left without any useful knowledge of any area. They may know what a transistor is, but they won't have any real idea of how to wire one up, or in what way computers are built from them. They may know the voltage changes in a Ethernet cable, but they don't know how to use that knowledge to improve their networks. It falls into a gap by being too technical for a casual reading book yet too impractical to be a technical book.

A Peek At Computer Electronics delivers exactly what is promised – a peek at the subject – and it does this well. It's engaging and there are plenty of diagrams to help explain the contents. However, this brief peek at electronics just isn't what we, as readers, want. We can't help but feel that most people would be better off with a book that focuses more on a single part of the subject area and covers it in enough detail to leave the reader with enough knowledge to be of practical use.

Rushes through too many topics to leave the reader with any useful information.

★★★☆☆



Yes, valves are covered in this book, despite not being used in computing for half a century.

Dart 1 For Everyone

Ben Everard steps up to the oche and learns a new language.

Author Chris Strom
Publisher Pragmatic Bookshelf
Price £15.99
ISBN 978-1-94122-225-6

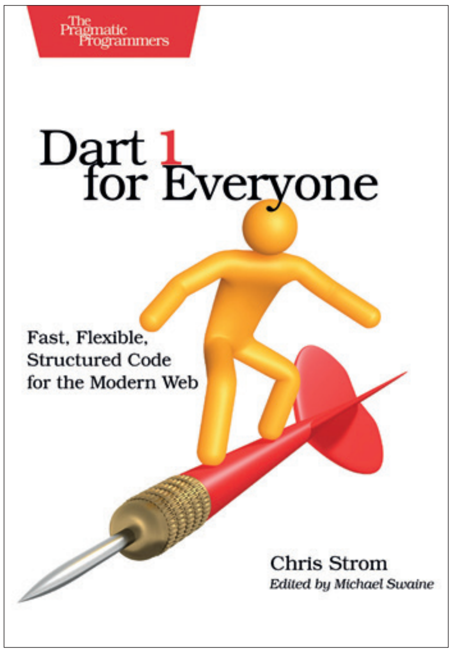
Let us be completely clear about one thing: *Dart 1 For Everyone* is not suitable for everyone. The book dives straight into programming and assumes that the reader can follow along without any explanation of how coding works. It doesn't even explain how Dart or JavaScript scripts run inside HTML in the browser. This book would be far better titled Dart 1 for Programmers, or better still, Dart 1 for JavaScript Programmers. With this caveat out of the way, let's take a closer look.

Dart is designed for building web applications, and *Dart 1 For Everyone* wastes no time in showing off its power in this regard. In the very first chapter, the book dives into a web front-end for a database that grabs the information it needs via AJAX. This leads us to the second caveat: it's entirely focussed on front-end development, yet many of the projects also need a back end, and that's not covered at all despite Dart having the capability to run on the server. Perhaps, then, the book should be titled *Dart 1 Web Frontends for JavaScript Programmers*, but I suppose that's a lot less snappy. That's the end of the caveats.

Dart 1 For Everyone particularly focuses on the aspects of Dart that are different to JavaScript (such as manipulating the DOM) and on the aspects that are important for modern web apps (such as the Model-View-Controller (MVC) architecture).

A good introduction to Dart 1, but makes promises that it doesn't keep.

★★★☆☆


The Pragmatic Programmers
Dart 1 for Everyone
Fast, Flexible, Structured Code for the Modern Web
Chris Strom
Edited by Michael Swaine

This book contains no useful advice on getting an elusive 180.

Hello Ruby

Graham Morrison rants a little at a crowdfunding prodigy.

Author Linda Liukas
Publisher Macmillan
Price £11.23
ISBN 978-1250065001

Hello Ruby started life as a Kickstarter campaign just as Linux Voice completed its own Indiegogo project in January 2014. It was hugely successful, with thousands of backers pledging a total of $380,747 to turn Linda Liukas' dream into reality. With hindsight, the project's success is perhaps no surprise. Linda did an amazing job at selling the idea of a children's book that could teach programming fundamentals through illustrations kids would respond to and a story more in common with Julia Donaldson than Bjarne Stroustrup.

But the book's journey from idea to reality has been a difficult one, especially for many of its backers (and we enthusiastically backed the project at the time). The book is over a year late, with many complaining their children are now too old for the narrative. More importantly, the book now has the backing of a major publisher, Macmillan. This has meant Kickstarter backers and Amazon pre-orderers are getting the book at almost the same time, yet Amazon's customers pay far less than the $60 we pledged as Kickstarter backers.

Continue rant

We understand crowdfunding and the huge challenges involved in getting things published. But it has been very difficult getting over the feeling that we, as backers, have subsidised what should have been Macmillan's investment in the title, with money that could have gone to other projects supporting children and technology. We're disappointed that neither Linda nor Macmillan seem to be addressing a problem that we feel will only impact similar worthy crowdfunding campaigns in the future.

The book itself is beautifully illustrated, with enough detail on each page to keep children amused and pointing at things. It's a simple story that follows a fierce little girl called Ruby as she takes on some challenges set for her by her father. There's nothing specific about programming, or even about technology, other than character names like Tux and Django, but the plot is constructed in such a way that Ruby has to solve problems and create solutions, useful skills for both programming and every day life.

The book ends with an activity section, originally promised as a separate workbook in the Kickstarter campaign. This involves putting instructions together and does introduce lots of programming terminology and ideas, testing the reader to come up with their own solutions in a form of pseudo-code. It works well, but there's a sharp jump in difficulty from the reading of the story to the challenges of the workbook.

Ignoring the negativity, Hello Ruby is a successful blend of storytelling, illustration and challenges. As the product of a dedicated individual who has put all this together, it's still hugely impressive. But as the work of a major publishing company, we can't help feel a little short-changed.

HELLO RUBY
Adventures in Coding
The world's most whimsical way to learn about COMPUTERS, PROGRAMMING, and TECHNOLOGY
Activities Included for all future coders
Linda Liukas
KICKSTARTER COPY: THANK YOU FOR YOUR SUPPORT!

Hello Ruby is about ideas rather than programming specifics.

As a crowdfunding campaign, it's slightly depressing. As a book, it's rather good.

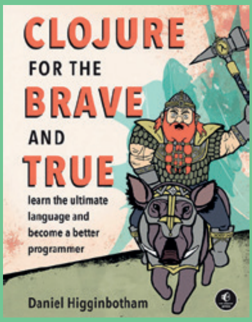
★★★☆☆

Also released...

November 2015

Clojure for the Brave and True

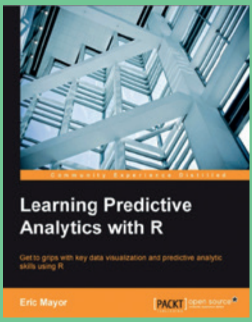
We admit, we chose this book purely on the strength of its cover. It looks like a demonic Viking riding a wild boar, of course. But it also made us realise that we know very little about the Lisp-based Clojure programming language other than how its pronounced (like closure), so perhaps we should read the words as well as look at the nice pictures.


CLOJURE FOR THE BRAVE AND TRUE
learn the ultimate language and become a better programmer
Daniel Higginbotham

Nobody tosses a Dwarf. Or, indeed, a Viking.

Learning Predictive Analytics

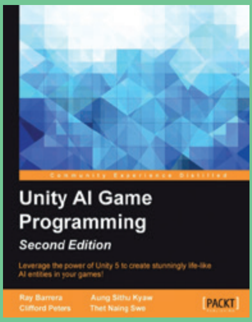
If you enjoyed this month's tutorial on statistical analysis with the GNU R programming language (see page 84), there's an entire book on the same subject, using the same language. It's going to be especially useful when you start downloading all that open data from your government, calculating MPs expenses and the rise in rail fairs.



Turn to p84 to see if Linux survives without Linus.

Unity AI Game Programming

Now that we have a native Linux version of the Unity games engine editor (see page 44), we need to prove that this porting effort was worth the trouble by creating awesome games on Linux. If only we knew how. Maybe we could code some AI that will do the job for us, then we can hire Linux Voice TNG and retire to Monaco with Jenson Button.



The *Unity* games engine is not the *Unity* desktop.

GROUP TEST

Graham Morrison takes a look at a range of software that may be slightly more productive for children than hours spent watching ukulele videos on YouTube

On test

Scratch

URL https://scratch.mit.edu
Licence GPLv2
Latest Release 1.4/2.0 (web)
The best-known tool for teaching children how to program.

TuxPaint

URL www.tuxpaint.org
Licence GPL
Latest Release 9.22
The Linux equivalent to Deluxe Paint is simple artistic fun.

TuxMath

URL http://tux4kids.alioth.debian.org
Licence GPL
Latest Release 2.0
Kid-friendly graphics and sounds help gamify simple maths lessons.

Sonic Pi

URL http://sonic-pi.net
Licence MIT
Latest Release 2.7.0
Used by serious musicians, it's also a great way to learn programming.

GCompris

URL http://gcompris.net
Licence GPL
Latest Release 15.02
This is the one to beat. Over a hundred mini-games and a framework for adults to supervise children's development

Childsplay

URL www.childsplay.mobi
Licence GPL
Latest Release 2.6.3
Another suite of activities, which can log statistics to an external database.

Educational software

Having spent half a lifetime playing video games, we'd argue that there's educational value in almost any kind of software – even *SuperTuxKart*. But for those of us with responsibilities to the young folk, finding something a little more brain nourishing than reflex attunement is a good way of introducing computing technology, Linux and learning, before letting your little ones run free with your pile of Gentoo install discs. It's also a great way of discussing some of the ideas behind open source, especially when so many children are now used to the freemiuum business model (free to download; pay through the nose if you want to get anything out of it) thanks to the apps they play on tablets and smartphones.

But creating a meaningful comparison of education software is difficult. This is because there isn't enough software to focus on a single age or schooling range, and also because the software that is available is rather disparate.

There are applications that deal with a single category, for example, such as astronomy or geography, while there are also tools that aim to provide an entire suite of learning, cramming hundreds of activities into a single application. Then there are the programming environments, or even flash cards, that are completely open and versatile to any use.

To side-step these issues while hopefully still providing a great overview of the software available, we're going to focus our comparison on whether an application succeeds in engaging its target audience while being able to teach its users effectively. That's a tough challenge, but equipped with some real-life young people (see How We Tested boxout, below) and a fresh Ubuntu installation, we think we can give it a jolly good go.

We'd argue that there's educational value in almost any kind of software – even Super TuxKart

How we tested

It wouldn't make much sense testing all this software ourselves. Out collective brain cells are too old, too cynical and too few. And as all this wonderful is software designed for the delicate sensibilities of The Next Generation, we'd be entering premature senility if we didn't put it to the test in front of its target audience. Over the course of a couple of weeks (and months, for some of our choices), we sat with a small group of children aged 4-10 and let them play with the software we showed them, noting their comments and allowing their feedback to shape our own conclusions. It wasn't a large enough group to allow scientific rigour, but it gave us a great idea of what worked, and what didn't.

Distros for children

Is it worth installing a distribution specifically designed for young people?

There are several Linux distributions designed specifically for children. Qimo and Edubuntu are both worth a look, as they each take different approach for making a computer more friendly and accessible. Qimo is more suitable for younger children, as its artwork and styling is designed to get kids clicking, but development has been slow recently. Edubuntu is better for older children, as it avoids them having to install a load of software before they can get started.

However, what we've found is that it's usually much more practical to install a standard Linux distribution, such as Mint, Ubuntu or Fedora, and let your children guide you in how they want the configuration to be adapted. You then have complete control over which software is installed and how the computer is used. All the most popular education applications and tools will be a few clicks away, and you can also make sure the machine is locked down adequately – running through a *Dansguardian* internet proxy filter, for example, and with YouTube restrictions in place.

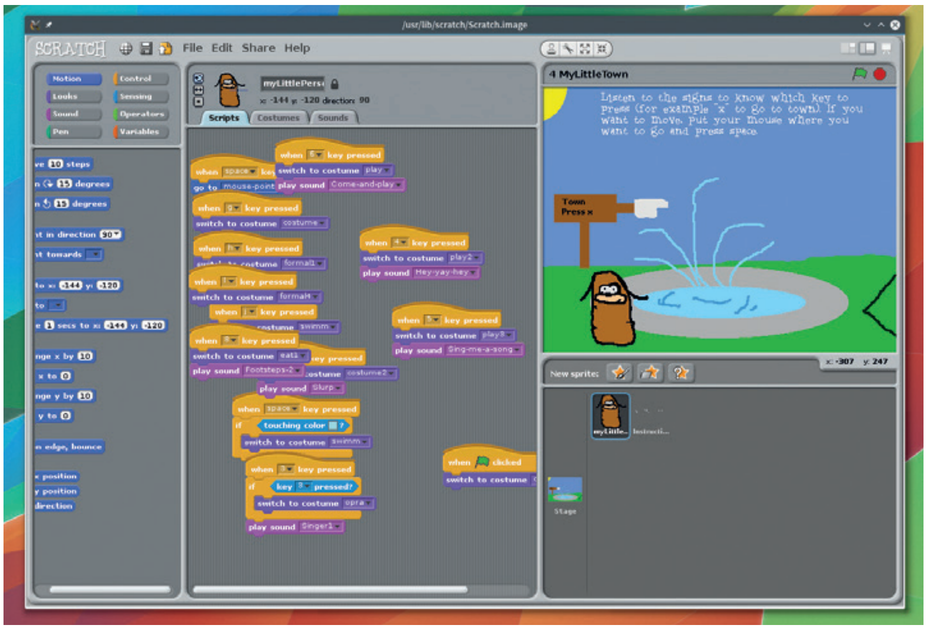
Scratch

Itching to get started?

Getting started with programming is difficult. But it's also vital. Code is the agent that binds computing to computing science. Even for students who have no intention of taking their studies further, unlocking this particular black box is the key to understanding how technology governs our lives, from ATMs to encryption. It's the equivalent of studying DNA in Biology, or the Periodic Table of Elements in Chemistry, or gravity in Physics, and we think programming should be a prerequisite for any computing course, regardless of age. This is something that's only going to become more important. The huge challenge, of course, is how best to teach programming, and Scratch is one of best attempts we've seen.

Scratch is a visual programming environment. A simple script-like syntax is augmented by colourful blocks that encase the various variables and keywords. The colour of each block is used to denote what each command is doing – purple is for visual changes such as output, for example, whereas dark orange is for events. Most things you write in Scratch are going to be driven by events, whether that's keyboard input or waiting for a value to be achieved. The jigsaw-like design of each block makes it easy to see the organisation, level and flow of the code, as well as which blocks are wrapped within others.

All this means that you learn exactly the same concepts as you would with a programming language like Python, without having to memorise lots of keywords or type too much, and apply them just as you would with any other language.



/usr/lib/scratch/Scratch.image
SCRATCH
File Edit Share Help
Motion
Control
Looks
Sensing
Sound
Operators
Pen
Variables
myLittlePers
Scripts
Costumes
Sounds
MyLittleTown
Listen to the signs to know which key to Press (for example 'x' to go to town). If you want to move, put your mouse where you want to go and press space.
Town Press x
New sprite:
Stage

Scratch projects can also run online with Flash, and the project's repository prepares young developers for ideas like forking and version control.

The visual style is also much easier to understand than abstract source code. But the best thing about Scratch is the vast amount of support material you have access to. This is partly thanks to the sterling work of the Raspberry Pi Foundation, which has pushed Scratch and even toyed with the idea of its devices booting into the environment in the same way Acorn machines booted to BBC BASIC, but also through necessity. The new UK national curriculum for 5–7-year-olds includes abstraction logic, algorithms and data representation, as well as writing 'computer programs' to solve problems, and Scratch is going to be perfect for this.

However, in practice, we have found that while Scratch is a wonderful environment for potential programmers, it doesn't do enough for self-learners, or perhaps for younger programmers. For us, the biggest challenge has been less about learning and more about finding something to inspire learning, and Scratch doesn't help with this. Just like any other language, it's there to get a job done, but it's up to you to find a job that will inspire your charges.

And this is the biggest problem with both learning to program and Scratch; you need to find the purpose that drives the learning. That said, as long as you're comfortable devising a program of study (loops, lists, control flow etc are all easy to implement) Scratch has tons of learning potential.

VERDICT

Wonderful as part of a project, but older children may be better off diving straight into Python.
★★★★☆

Tux Paint

It's just like painting, only without the mess on the floor and the tidying up.

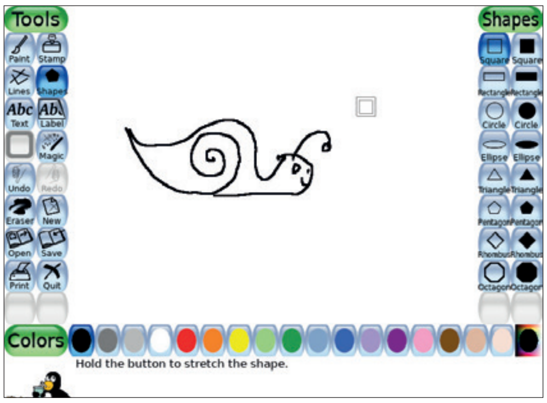
Anyone with kids and a smartphone knows that the first thing they want to do is paint onto your screen. And there's a good argument to suggest that the Amiga's initial success was partly because Commodore bundled the brilliant drawing package, *Deluxe Paint*. The addictive pleasure of multi-coloured point-and-click sketching was a great motivator for mastering the mouse. And it's the same for *Tux Paint*. Despite its toddler-friendly icons and sound effects, we've found that its compulsive drawing engine has never failed to entertain children under 10. They just love mucking about, much as they do with an Etch A Sketch.

Paint

Tux Paint seems to take inspiration from *Deluxe Paint*. The three borders of the canvas area where you paint are used for choosing a drawing tool, selecting a shape or a brush and then a colour. Thanks to the icons, a three-year-old can pick up the process quickly, and while the sounds are initially appealing, they quickly get annoying (for adults, anyway). The application also defaults to full-screen, which works well for most, but we couldn't get window mode working without a configuration file.

The stamps are good fun, but we'd love to see the simple foreground/background layers that *Deluxe Paint* had, as well as a similar animation engine – effectively a fast slideshow. *Deluxe Paint* could also animate the brushes (sprites), which made creating animations as fun and as easy as drawing. Features like these would help make *TuxPaint* more than a simple distraction, and help potential artists and animators learn more skills than simple mouse control.

Thanks to Tux Paint's icons, a three-year-old can pick up the process



Drawing is fun and helps children learn to use the mouse.

VERDICT
Good for encouraging use of a mouse, but limited in its educational potential.
★★★☆☆

TuxMath

Maths makes sense.

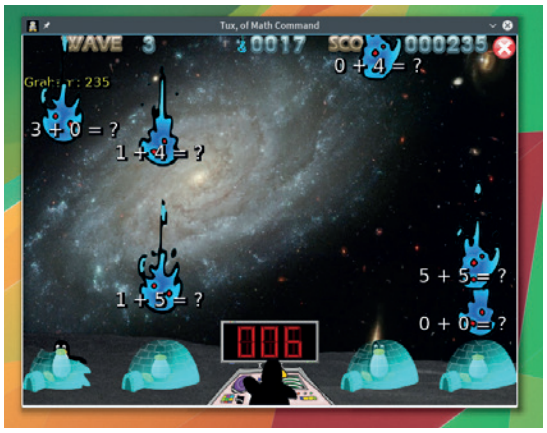
TuxMath is similar to *Tux Paint*, thanks to the project being largely from the same development team and sharing the same cartoony style and audio. But it takes learning further, and would make a good next step for young children needing something more practical. We found that when they're old enough to have had some preliminary exposure to numbers and simple addition, perhaps 3–5 years old, *TuxMath* was both entertaining and helped to build their confidence with numbers.

Through a series of mathematically themed games, *TuxMath* becomes far more than a distraction, often combining quick thinking with keyboard and mouse skills. The games themselves are variations on the arcade classic *Missile Command*, where you stop the imminent collision of a meteor by typing in the answer to a sum displayed across each descending foe. Typing the answer and pressing return will shoot a laser beam to vaporise any threats whose answer you've hit.

Levels of challenge

There are many difficulty levels to choose from, from simple addition through to the division of positives and negatives (-10/2, for example), and the objects' descent gets quicker and quicker, turning the game into quite a challenge. Even older kids will get a kick out of beating their high scores as the game itself becomes quickly addictive.

There's a shared screen multiplayer mode, which could do with a handicap setting for one player, and a client/server mode that can be used to dispatch the same lessons across a network of machines, making it ideal for a workshop or club. We find the music slightly incongruent with mathematical study, especially as it loops back to the beginning after a few minutes, but sound can be disabled from the command line.

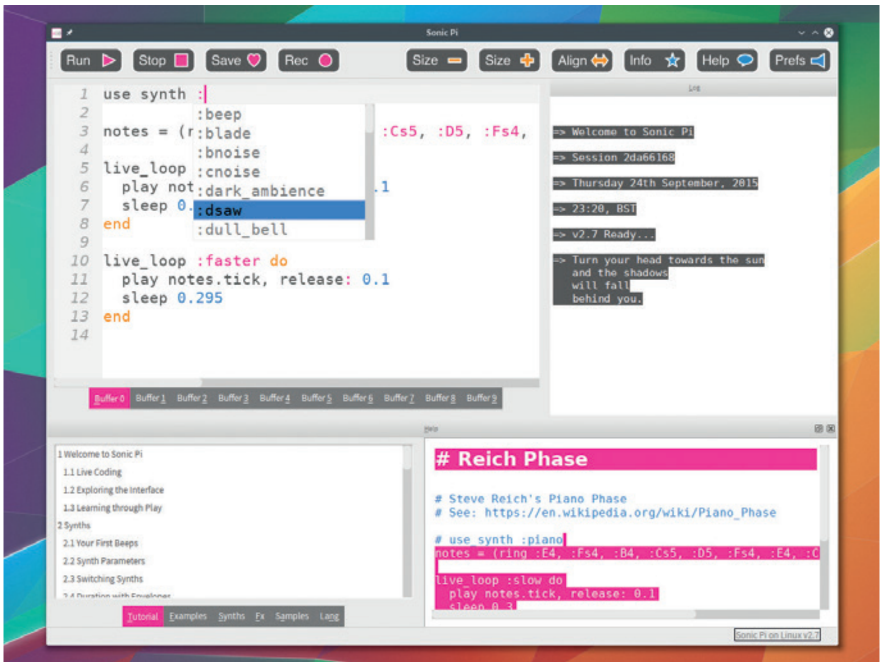


TuxMath includes a version of *Asteroids* where large asteroids break into their factors.

VERDICT
Gameplay and learning potential make up for the lack of in-game options.
★★★½☆

Sonic Pi

Learn to code and write music all at the same time.



```
use synth :
notes = (r          :Cs5, :D5, :Fs4,
live_loop
  play not          .1
  sleep 0.
end

live_loop :faster do
  play notes.tick, release: 0.1
  sleep 0.295
end
```

```
=> Welcome to Sonic Pi
=> Session 2da66168
=> Thursday 24th September, 2015
=> 23:20, BST
=> v2.7 Ready...
=> Turn your head towards the sun
   and the shadows
   will fall
   behind you.
```

```
# Reich Phase

# Steve Reich's Piano Phase
# See: https://en.wikipedia.org/wiki/Piano_Phase

# use_synth :piano
notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4, :E4, :C

live_loop :slow do
  play notes.tick, release: 0.1
  sleep 0.3
```

The flexibility in Sonic Pi comes from the powerful SuperCollider audio programming system.

When it comes to technology, the thing that often comes after messing about with virtual paint is music and sound. Whether that's mashing the keys on a piano or banging a drum, there's something special about its spontaneity. Instant feedback and familiarity make a great teacher, and the formulaic nature of audio generation and music composition makes audio perfectly suited to programming. Sonic Pi capitalises on this, presenting the user with a real-time programming environment that's capable of some serious results. Its creator, Dr Sam Aaron at the University of Cambridge, live-codes Sonic Pi with his band, and there are many complete algorithmic compositions to download as examples.

Sonic Pi is also easy to use. At its simplest, you can play a pitch by typing **play 70** and clicking on Run in the toolbar. The syntax is always simple, and the exceptional inline help and tutorials system make it easy to progress without any external resources. The loop-based nature of modern music lends itself well to common structures like blocks, arrays, functions and iteration, and the many sound generators and variables provide huge scope for experimentation. Even advanced topics like threads find their way into Sonic Pi, as they're used to play different sections of code simultaneously.

Sonic the leap-frog

This level of coding is obviously going to be most beneficial to older children, but we found the audio feedback much more natural than the graphical style of Scratch, and because you can change your code while it's playing back, you can see how different parameters and keywords affect the feedback. For our kids, it added self-propelled exploration, despite the typing and text involved in writing code. Even younger children had fun changing octaves and synth parameters, gaining an insight into the types of variables and their ranges, as well as what computers do behind their screens.

The only negative with Sonic Pi is that because it's partly funded by the Raspberry Pi Foundation and is pre-configured to run on a Pi, it's a little clunky on a desktop. It also needs to run through the Jack audio layer, which the average Linux geek will have problems getting to run correctly, let alone a 10-year-old.

VERDICT

This might be the best way to get musical kids into programming.

★★★★★

Online resources

The only reason to keep Flash installed.

Although we haven't included them, there are dozens of online resources and educational games to play online with children. Scratch 2.0, for example, can be used, shared, modified and edited from its web portal without installing a single package. This could be a great option if you're teaching kids using a variety of operating systems, or if they want to access their projects from home. The BBC also hosts some wonderful content, from games through to study guides. For pre-school children, its CBeebies activities page is essential (**www.bbc.co.uk/cbeebies**), as too is the free Android and iOS app.

For older children, the BBC has a comprehensive set of resources that covers the national curriculum from key stage 1 through to GCSEs. The Raspberry Pi Foundation has also done a brilliant job with its education resources (**www.raspberrypi.org/resources**), turning the creation of hardware into its own entertainment, from teachers to makers. Even adult resources like Khan Academy (**www.khanacademy.org**) or many of the TED talks (**www.ted.com/talks**) can provide great insight into a subject, as too can many of the free audio and ePub books you can download through projects like Project Gutenberg (**www.gutenberg.org**). The only problem is that many of these portals and games require Flash, which we're loath to recommend, and your children will obviously need supervision and support when left to the wilds of the internet.

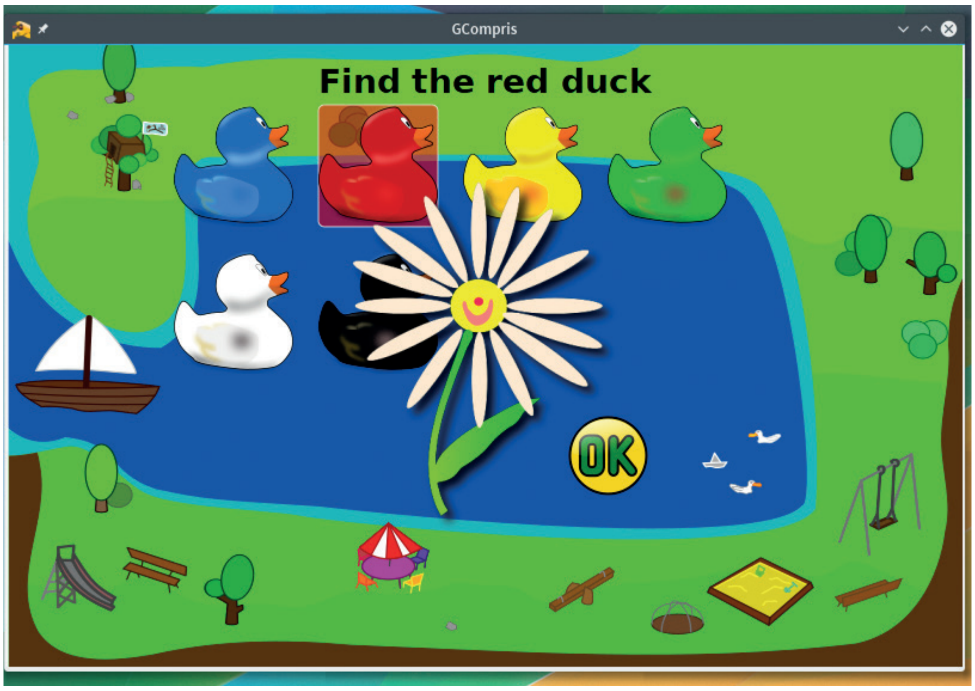
We hope the BBC follows-up its iPlayer migration from Flash by porting its ace edutainment too.

GCompris vs Childsplay

Two activities suites fight it out to control the mind.

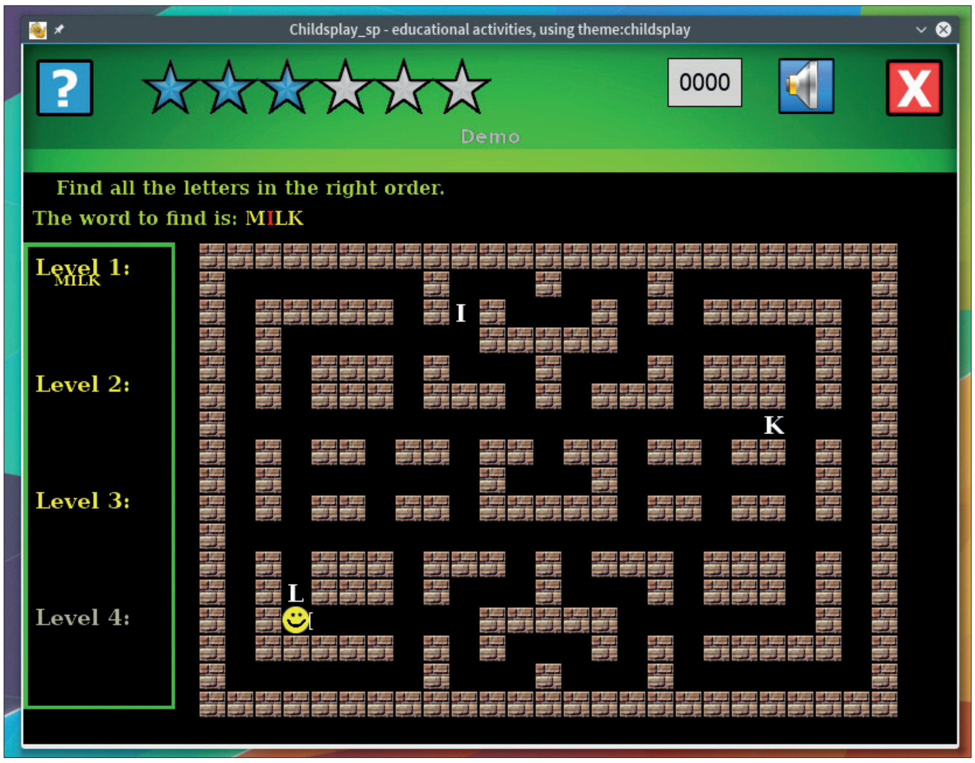
When it comes to educational software for young children, *GCompris* is often considered the go-to package to install. It's tailored specifically for young children – perhaps as their first experience with computers, and it bundles so many mini-games that you could dedicate an old computer to running *GCompris* and nothing else.

The games themselves are split into various categories, and are navigated using a simple folder system. Many of the games are simple, but the ideas behind them are clever. One of the early activities has the player pressing both Shift keys at the same time, for example, while more activities practice reading or playing strategy games like chess or Mancala. Our test audience didn't always respond well to the colourful and brash aesthetics, and the simple animations don't compare to the smooth transitions found on tablets and smartphones. We'd also like to see a button on each screen that repeats the instructions for each game, as the attention spans of small children often mean they miss the description.



The graphics may be simple, but with almost 150 activities, there's hours of fun to be had in *GCompris*.

The best thing about *GCompris*, however, is the management interface. This is designed for an adult and it enables you to see exactly what your children have been doing. The time they spend on each game is logged and listed, along with which levels they played. You can also enable and disable certain titles, and create group and class categories to better track the children playing the games.


Childsplay_sp - educational activities, using theme:childsplay
0000
Demo
Find all the letters in the right order.
The word to find is: MILK
Level 1:
MILK
Level 2:
Level 3:
Level 4:
I
K
L

Childsplay has less educational value, although getting children into *Pac-Man* via spelling is genius.

Here's Chucky

Childplay is another application that offers a suite of activities for children. It can't compete with *GCompris* when it comes to the number on offer, but its design is slightly slicker and you don't have to make your first mission turning off the inane background music. There are fewer than 20 games, including flash cards and putting animal sounds to the images, and even a game of *Pong*. But unlike *GCompris*, there's no structure to the games, no parental control or management, and the activities are often simplistic. However, our kids had fun playing Simon Says and the *Pacman*-style spelling games, and it's definitely worth a try.

VERDICT

GCOMPRIS A huge suite of activities that succeeded in keeping younger wards quiet.
★★★★☆

CHILDSPLAY Too few activities, but it's worth installing as another option.
★★☆☆☆

OUR VERDICT

Educational software

None of us want our children to spend too much time in front of a screen. But computing has become such an important part of the world around us that getting them used to the technology is just another step in their educational journey.

Our small team wanted to learn, but the messages from the software were mixed. As wonderful as *GCompris* is, for example, it's hard to pitch it against the glossy apps that many children spend time with, and the disparate collection of 'brain training' activities are difficult to justify. For us, this means asking what you want educational software to achieve, and the best answer we have is that we want this software to teach about computers. And the language of computers is programming.

That means, for us, the winner of this test needs to be something that teaches programming. It's widely accepted that Scratch does a brilliant job at this, especially in a classroom. But we can't help thinking that Scratch doesn't make programming any easier, just because it's visual. What we found with our children is that Scratch doesn't inspire learning either, at least until you've got the skills to work on your own projects. This is why Sonic Pi wins. We loved the user interface and the way its baby-step tutorials were written and embedded within the main application. The application itself has developed so rapidly over the last 12 months, that it almost feels like a modern IDE. But best of all, sound production gives immediate feedback, inviting you to play with it and mess with the values.

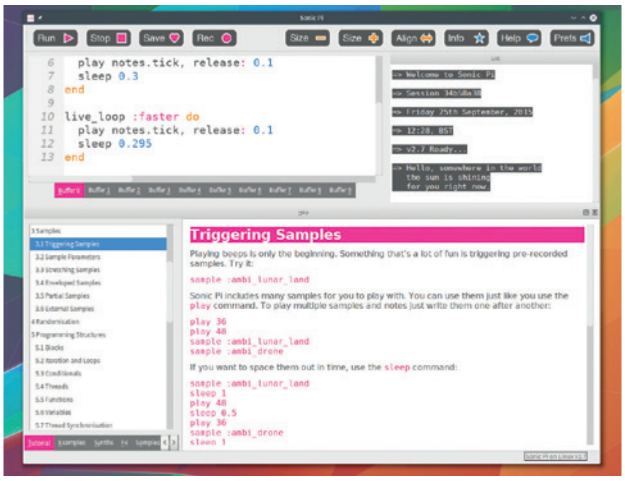
Music == code

Even if you don't care for music or sounds (and we've yet to meet a child like this), audio has evolved to give us this natural feedback, and coding both timbre and rhythm are brilliant ways of learning about writing functions and manipulating data structures. But best of all, it gives you a reason to try and to take your skills to the next step, all within a single application. And when you want to take your code into the hipster stratosphere, you're ready to take on Pure Data, which is a kind of visual programming environment that inadvertently teaches you all the subtlety of functional programming – hopefully before you've left school.

Sonic Pi has developed so quickly over the last 12 months that it almost feels like a modern IDE

Other applications

So much of the software we use day-to-day is perfect for children and their development. Online maps, for example, are better than any old geography book, and applications like *Blue Marble* bring all the wonder of the world to your desktop, allowing your kids to explore a virtual globe with just a mouse. *KGeography* is a brilliant application for testing their knowledge, as too is *Kalzium* for exploring the periodic table of elements. Leaving earth behind, *Stellarium* is a brilliant astronomy application that can do the same for space, and while *Celestia* development has stalled, we've yet to find another application that will let you fly from one star system to another in beautiful OpenGL-rendered graphics.



```
  play notes.tick, release: 0.1
  sleep 0.3
end

live_loop :faster do
  play notes.tick, release: 0.1
  sleep 0.295
end
```

```
=> Welcome to Sonic Pi
=> Session 34b5Ba38
=> Friday 25th September, 2015
=> 12:28, BST
=> v2.7 Ready...
=> Hello, somewhere in the world
   the sun is shining
   for you right now.
```

Triggering Samples

Playing beeps is only the beginning. Something that's a lot of fun is triggering pre-recorded samples. Try it:

```
sample :ambi_lunar_land
```

Sonic Pi includes many samples for you to play with. You can use them just like you use the play command. To play multiple samples and notes just write them one after another:

```
play 36
play 48
sample :ambi_lunar_land
sample :ambi_drone
```

If you want to space them out in time, use the sleep command:

```
sample :ambi_lunar_land
sleep 1
play 48
sleep 0.5
play 36
sample :ambi_drone
sleep 1
```

We loved Sonic Pi. If you're looking for an entry point to programming and creating sounds with instant feedback, it's the perfect experimental platform.

1st Sonic Pi

http://sonic-pi.net
Fun and serious at the same time, this could prove the inspiration for the career of your own fame-obsessed pop stars.

2nd GCompris

http://gcompris.net
Worthy of second simply because of the number and breadth of activities here. Also better suited for younger children.

3rd Scratch

https://scratch.mit.edu
We still love Scratch and it's capable of more than Sonic Pi, but you need to put more effort into the preparation to inspire your children to use it.

4th TuxMath

http://tux4kids.alioth.debian.org
Simple and effective at getting children to learn about numbers while they think they're being allowed to play video games.

5th TuxPaint

www.tuxpaint.org
This is still a brilliant application; creative, relaxing and good for teaching mouse control; it's just lacking any educational rigour.

6th Childsplay

www.childsplay.mobi
The few activities in *Childsplay* are fun, and worth an installation, but they won't keep anyone's attention going for as long as the other tools here.

Subscribe

shop.linuxvoice.com

Introducing **Linux Voice**, the magazine that:

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

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

Get 100 pages of tutorials, features, interviews and reviews every month

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

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

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

NEXT MONTH IN LINUXVOICE

ON SALE THURSDAY 19 NOVEMBER

RISE OF THE ROBOTS

Under the sea, in space, clearing minefields, looking for earthquake survivors, tracking down survivors of the nuclear holocaust – Linux-powered robots are everywhere!

EVEN MORE AWESOME!

Economic modelling

Predict the next catastrophic financial collapse with this open source, GPLed software. Interest rates have never been so much fun!

Microsoft!

We chew the fat with Gianugo Rabellino, Microsoft's senior director of open source programs, to find out what the Windows giant really thinks about Linux.

```
top - 09:53:26 up 24 min,  4 users,  load average: 0.73, 0.69, 0.76
Tasks: 210 total,   1 running, 209 sleeping,   0 stopped,   0 zombie
%Cpu0  :   9.7/7.6     12[
%Cpu1  :   5.0/1.3      7[
%Cpu2  :  12.3/1.3     14[
%Cpu3  :   3.3/2.0      5[
GiB Mem :  14.4/15.640    [
GiB Swap:   0.0/20.000    [

  PID USER      PR  NI    VIRT    RES  %CPU %MEM     TIME+ S COMMAND
    1 root      20   0   35.7m   5.0m   0.0  0.0   0:00.65 S systemd
  198 root      20   0   56.4m  24.2m   0.0  0.2   0:00.96 S  `- systemd-journal
  223 root      20   0   40.1m   5.0m   0.0  0.0   0:01.06 S  `- systemd-udevd
  526 dbus      20   0   38.0m   4.5m   0.0  0.0   0:00.38 S  `- dbus-daemon
  529 root      20   0   19.3m   2.8m   0.0  0.0   0:00.01 S  `- systemd-logind
  532 root      20   0   89.0m   4.3m   0.0  0.0   0:00.03 S  `- login
  592 graham    20   0   20.0m   3.4m   0.0  0.0   0:00.01 S      `- bash
  630 graham    20   0   17.8m   2.8m   0.0  0.0   0:00.00 S          `- startx
  652 graham    20   0   21.8m   7.1m   0.0  0.0   0:00.00 S              `- xinit
  653 root      19  -1  185.1m  83.7m   4.0  0.5   0:47.66 S                  `- Xorg
  662 graham    20   0   17.8m   2.8m   0.0  0.0   0:00.01 S                  `- start
  711 graham    20   0   88.4m   5.0m   0.0  0.0   0:00.00 S                      `- k
  574 root      20   0   43.6m   5.1m   0.0  0.0   0:00.00 S  `- sshd
  587 graham    20   0   37.2m   4.6m   0.0  0.0   0:00.02 S  `- systemd
  589 graham    20   0   96.6m   1.6m   0.0  0.0   0:00.00 S      `- (sd-pam)
  677 graham    20   0   39.8m   5.5m   0.0  0.0   0:01.07 S      `- dbus-daemon
  715 graham    20   0  176.2m   4.4m   0.0  0.0   0:00.00 S      `- dconf-service
  720 graham    20   0  447.4m  41.0m   0.0  0.3   0:00.33 S      `- kglobalaccel5
  768 graham     9 -11  485.1m  13.1m   0.7  0.1   0:05.69 S      `- pulseaudio
  834 graham    20   0  496.8m  20.0m   0.0  0.1   0:01.40 S      `- tracker-store
  845 graham    20   0  325.7m   6.0m   0.0  0.0   0:00.03 S      `- zeitgeist-daemo
  849 graham    20   0  209.7m   6.1m   0.0  0.0   0:00.01 S      `- gvfsd
  855 graham    20   0  289.2m   6.5m   0.0  0.0   0:00.00 S      `- gvfsd-fuse
```

Bash

Hack your Linux shell to get more power, more features, and spend more time walking the dog while your computer does the hard work for you.

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

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

FOSSpicks

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

Between sessions foraging for mushrooms and apples, **Ben Everard** snuffles through the forest looking for the best Free Software.

Web reader

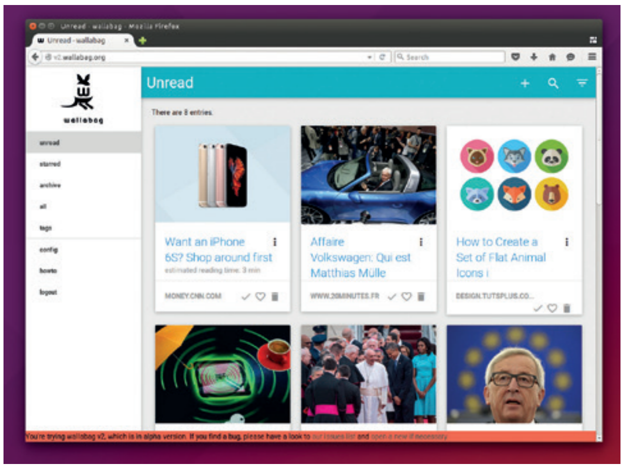
Wallabag

The web is made for browsing – but sometimes you come across something fascinating that you don't have time to read, and want to come back to it later. This is where *Wallabag* comes in. It's part bookmark manager for synchronising bookmarks between devices, part re-formatter and part download manager.

Wallabag links into your web browser, so that when you get to a page you want to save for later, you activate it. This saves the pages, and alters them to make them more readable. It can even transform them into a different format (such as ePub) that may be more suitable for your device. It's all open source, but you'll need to run part of it on a server and part on the clients you connect with.

Wallabag makes web pages available on all your devices when you want them, not just when you have an internet connection, and does so without using any proprietary software, so you don't have to surrender your data to the whims of advertisers.

PROJECT WEBSITE
www.wallabag.org



You can test out *Wallabag* without installing it at **http://v2.wallabag.org/** with the login **wallabag/wallabag/**.

Secure chat

Ricochet

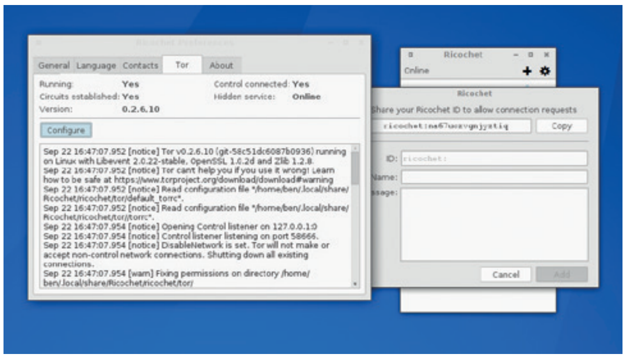
Ricochet (formerly known as *Torsion*) is an anonymous, secure messaging platform. On installing the software, you automatically get a username that looks like a random set of characters with the word 'ricochet' at the start. Using this, you can connect with other users to send and receive instant messages.

There are versions for Linux, Windows and Mac OS X, so you can talk to people who haven't yet discovered the joys of Linux as well as those who have. At the moment, *Ricochet* is limited to just text messages, but the protocol is extensible, and future releases may include the ability to send files and potentially video chat.

Anonymity is created by routing connections through the *Tor* network, and your *Ricochet* username is in reality just the address of a hidden service running on your machine. The anonymity is as good as *Tor*'s hidden services. This is good, but not perfect. It is likely that users will be identifiable by organisations with large resources, although this is still likely to be better than most options.

Given the young status of *Ricochet*, hidden services are unlikely to be the worst issue with the security. It takes time and a lot of eyeballs to create software that completely protects the users. For now, *Ricochet* is probably secure enough for those who don't want the state spying on them routinely, but who don't have anything that could be considered high-value.

Ricochet is limited to just text messages, but the protocol is extensible



Running: Yes
Circuits established: Yes
Version: 0.2.6.10
Control connected: Yes
Hidden service: Online

```
Sep 22 16:47:07.952 [notice] Tor v0.2.6.10 (git-58c51dc6087b0936) running
on Linux with Libevent 2.0.22-stable, OpenSSL 1.0.2d and Zlib 1.2.8.
Sep 22 16:47:07.952 [notice] Tor can't help you if you use it wrong! Learn
how to be safe at https://www.torproject.org/download/download#warning
Sep 22 16:47:07.952 [notice] Read configuration file "/home/ben/.local/share/
Ricochet/ricochet/tor/default_torrc".
Sep 22 16:47:07.952 [notice] Read configuration file "/home/ben/.local/share/
Ricochet/ricochet/tor//torrc".
Sep 22 16:47:07.954 [notice] Opening Control listener on 127.0.0.1:0
Sep 22 16:47:07.954 [notice] Control listener listening on port 58666.
Sep 22 16:47:07.954 [notice] DisableNetwork is set. Tor will not make or
accept non-control network connections. Shutting down all existing
connections.
Sep 22 16:47:07.954 [warn] Fixing permissions on directory /home/
ben/.local/share/Ricochet/ricochet/tor/
```

Share your Ricochet ID to allow connection requests
ricochet:na67uoxvgnjyztiq
ID: ricochet:
Name:
Message:

Protect yourself from unwanted eavesdropping with *Ricochet*'s *Tor*-based web chat.

PROJECT WEBSITE
https://ricochet.im

Map editor

iD

The Wikimedia Foundation has just launched a new map service that takes the data from OpenStreetMap and renders it in a format suitable for Wikipedia and other resources. This is a fantastic idea, so we decided to take a look at OpenStreetMap.

We came across *iD*, which is the web-based editor for OpenStreetMap. It's open source, but you don't need to do anything to install it, just point your browser to **www.openstreetmap.org**, and you should find the Edit tab in the top-left corner of your screen. Clicking on this will open the *iD* editor and enable you to modify the map directly from your browser.

You will need to create a login first, but this doesn't try to harvest too much data (just a name and email address), or you can log in using a social media account.

The first time you sign in, you'll be invited to follow a walkthrough that will give you a reasonable idea of how to use the software to edit maps. It's straightforward for basic editing and is ideal if you want to add a point of interest, or add information to items that are already on the map.

Open data

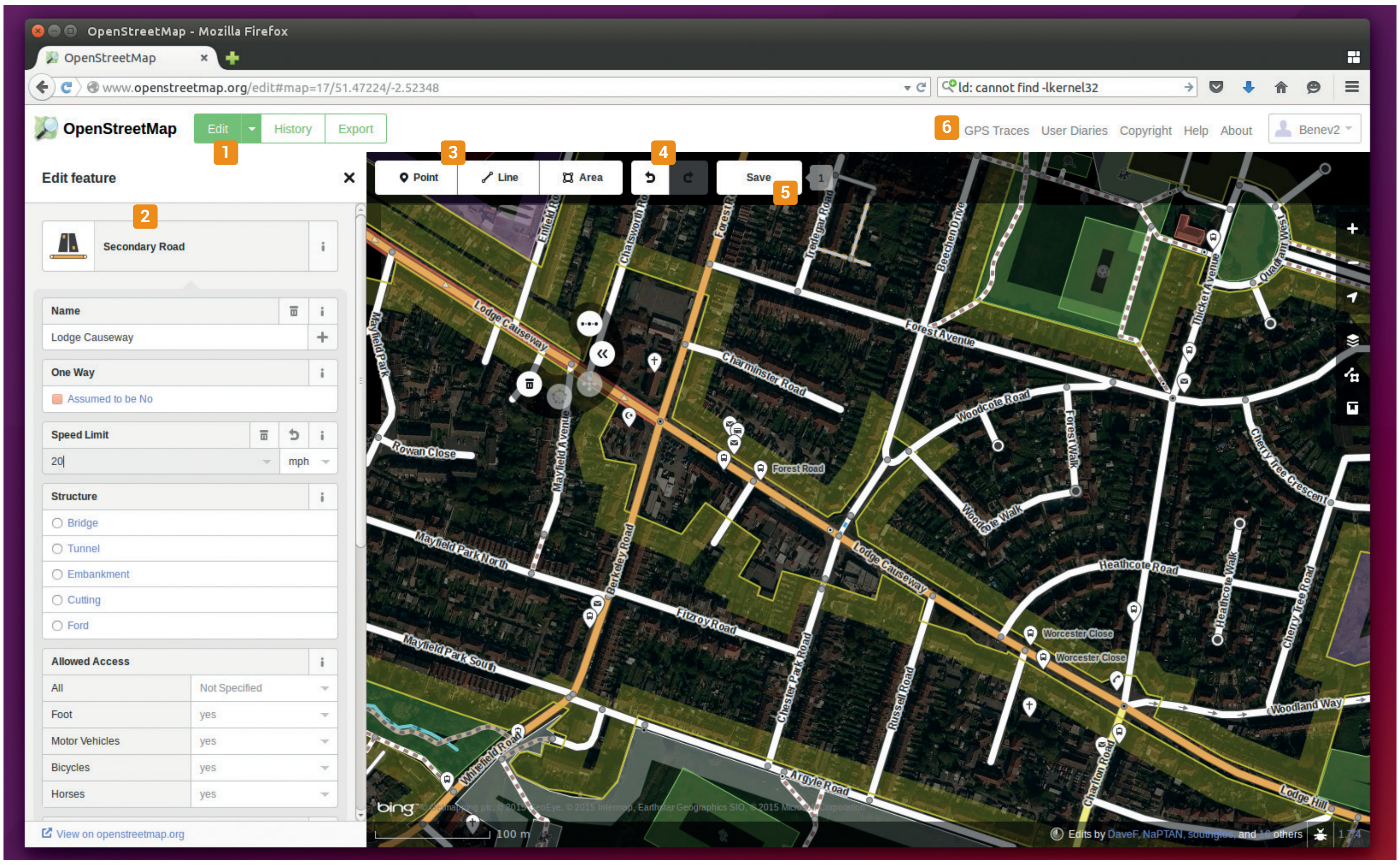
Maps based on this data are produced by rendering the raw information into images. Different mapping services use different rendering processes, so your changes may take some time to appear in a particular set of maps.

If you're doing significant edits to the map (for example, using a GPS trace to alter or add a road), then it's probably better to use a more powerful piece of software, such as *JOSM*. However, now that the maps are already quite complete, there are fewer major edits left to do. Gathering all these little edits, such as points of interest and details of buildings, is an important task in building up the maps.

Open map data is important because it enables anyone to create geo-aware applications and services. If we allow global corporations to gather the most accurate maps of the world, we allow them to dictate the terms with which we see our environment.

The beauty of *iD* is that there's nothing to install, and it doesn't take much technical skill to use. If you're looking for a way to give back to the free software community, but you're not a programmer, open mapping with *iD* is a fantastic option.

PROJECT WEBSITE
www.openstreetmap.org


OpenStreetMap - Mozilla Firefox
OpenStreetMap
www.openstreetmap.org/edit#map=17/51.47224/-2.52348
Id: cannot find -lkernel32
OpenStreetMap
Edit
History
Export
GPS Traces
User Diaries
Copyright
Help
About
Benev2
Edit feature
Point
Line
Area
Save
1
2
3
4
5
6
View on openstreetmap.org
Edits by DaveF, NaPTAN, southglos and 10 others

Secondary Road

Name: Lodge Causeway

One Way: Assumed to be No

Speed Limit: 20 mph

Structure: Bridge, Tunnel, Embankment, Cutting, Ford

Allowed Access

All	Not Specified
Foot	yes
Motor Vehicles	yes
Bicycles	yes
Horses	yes

1 Edit tab 2 The features sidebox enables you to change the attributes of the currently selected item 3 Select the type of item you wish to add to the map 4 Undo and redo changes 5 Save and upload your changes to OpenStreetMap 6 GPS traces are information added by people who have physically travelled the routes.

Emulator

Angel

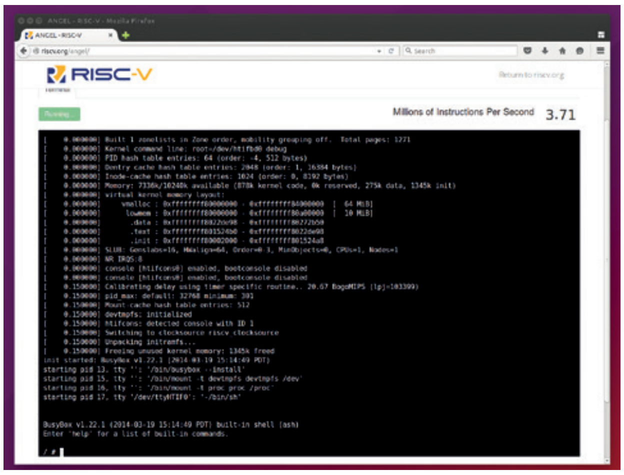
ANGEL Is Naturally Good at Executing Linux is the recursive name for this JavaScript implementation of the RISC-V instruction set. In other words, it's an emulator that can run in your web browser. Just head to **http://riscv.org/angel** and you can boot up a Linux session. It should run in any modern browser, but we found that it ran about four times faster in *Chromium* than *Firefox*. The session is fairly minimal, running little more than the Linux kernel and the BusyBox shell. There's not even network support, so it's quite hard to get any more advanced software running.

Angel isn't the first emulator to run in JavaScipt. JSOR1K (JavaScript OpenRisc 1K) is an implementation of the OpenRisc 1K instruction set that you can boot by pointing your browser to **https://s-macke.github.io/jor1k/demos/main.html**. This emulator is a little more useful, since it has more utilities and even network access, so you should be able to download and compile software should you need it.

Linux in a browser

It's hard to say whether these emulators have any real value beyond novelties. They could be used to port terminal applications to the web, or they could be used to help people get started with Linux in a safe environment.

In reality though, the main reason for these is that there's something cool about running a full operating system inside your browser. It's over the top, it's geeky beyond words, it makes no sense from a practical point of view, and we love it because of this.



```
[    0.000000] Built 1 zonelists in Zone order, mobility grouping off.  Total pages: 1271
[    0.000000] Kernel command line: root=/dev/htifbd0 debug
[    0.000000] PID hash table entries: 64 (order: -4, 512 bytes)
[    0.000000] Dentry cache hash table entries: 2048 (order: 1, 16384 bytes)
[    0.000000] Inode-cache hash table entries: 1024 (order: 0, 8192 bytes)
[    0.000000] Memory: 7336k/10240k available (878k kernel code, 0k reserved, 275k data, 1345k init)
[    0.000000] virtual kernel memory layout:
[    0.000000]     vmalloc : 0xffffffff80000000 - 0xffffffff84000000   [  64 MiB]
[    0.000000]      lowmem : 0xffffffff80000000 - 0xffffffff80a00000   [  10 MiB]
[    0.000000]       .data : 0xffffffff8022de98 - 0xffffffff80272b50
[    0.000000]       .text : 0xffffffff801524b0 - 0xffffffff8022de98
[    0.000000]       .init : 0xffffffff80002000 - 0xffffffff801524a8
[    0.000000] SLUB: Genslabs=16, HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS:8
[    0.000000] console [htifcons0] enabled, bootconsole disabled
[    0.000000] console [htifcons0] enabled, bootconsole disabled
[    0.150000] Calibrating delay using timer specific routine.. 20.67 BogoMIPS (lpj=103399)
[    0.150000] pid_max: default: 32768 minimum: 301
[    0.150000] Mount-cache hash table entries: 512
[    0.150000] devtmpfs: initialized
[    0.150000] htifcons: detected console with ID 1
[    0.150000] Switching to clocksource riscv_clocksource
[    0.150000] Unpacking initramfs...
[    0.150000] Freeing unused kernel memory: 1345k freed
init started: BusyBox v1.22.1 (2014-03-19 15:14:49 PDT)
starting pid 13, tty '': '/bin/busybox --install'
starting pid 15, tty '': '/bin/mount -t devtmpfs devtmpfs /dev'
starting pid 16, tty '': '/bin/mount -t proc proc /proc'
starting pid 17, tty '/dev/ttyHTIF0': '-/bin/sh'


BusyBox v1.22.1 (2014-03-19 15:14:49 PDT) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ #
```

Get a full Linux command line in your browser with a JavaScript RISC emulator.

Angel is over the top, geeky beyond words, makes no sense, and we love it

PROJECT WEBSITE
http://riscv.org/angel

Sandbox control

Firejail & Firetools

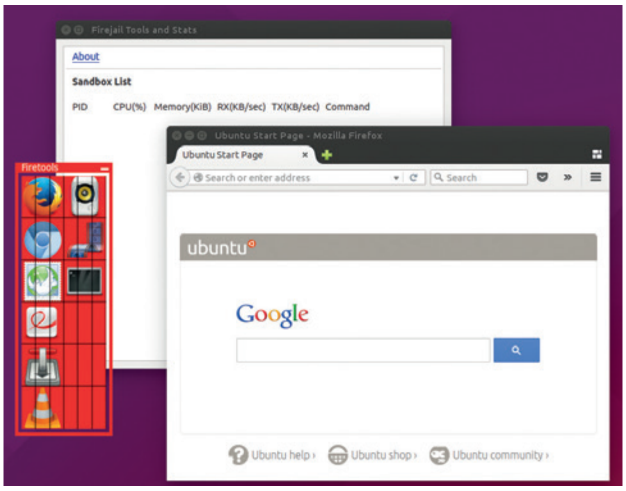
When you launch an application in Linux, the software usually gets the same privileges on the system that you have. For example, when you run *Firefox*, it gets permission to access the files in your home directory. Usually, this is a good thing. It means that you can access your files and upload them to websites, should you wish. However, it also means that if an attacker is able to subvert *Firefox*, they may also get access to your home directory.

This is exactly what happened in August 2015. A bug in the **pdf.js** viewer allowed attackers to scan victims' home folders and steal the contents of the **.ssh** folder, which potentially gave them access to any remote machines that the victim had access to.

The solution to this is to run software in a restricted environment where it can't access the main system. *Firejail* is a piece of software that is used to launch other pieces of software in limited environments. You can open the web browser through *Firejail* with the following (you must have closed all browser sessions beforehand):

```
firejail firefox
```

Firefox is now running in a restricted environment and can't access the **~/.ssh** directory (try to open this directory through the open file dialog and you'll get a permissions error). There's a specific *Firefox* profile that defaults to this level of protection because, really, your web browser has no business looking in that directory. This is just a simple example. There are loads of ways you can lock down the software you use, and there's even a GUI to make it easy in the form of *Firetools*.



Firejail can isolate any software, but the web browser gains the most since it's the most frequently attacked.

PROJECT WEBSITE
https://l3net.wordpress.com/projects/firejail/

Source code hosting

GitLab

If you're familiar with open source software, you're almost certainly familiar with the source code hosting platform GitHub. Even if you don't code yourself, you've probably found yourself getting software from it at some time or another. GitLab offers a very similar feature set to GitHub, but the entire setup is open source. This means that you can host it yourself rather than relying on a company to do it for you.

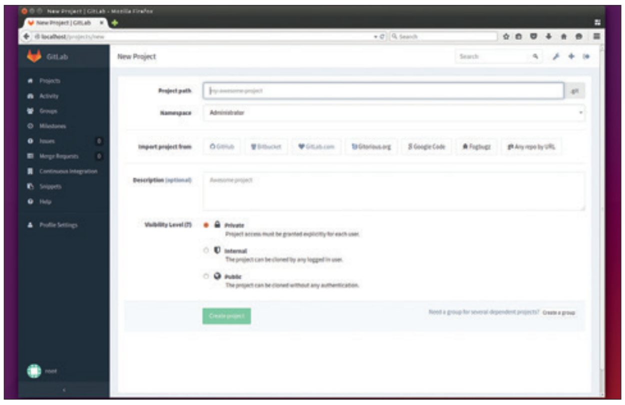
There's the web interface to *Git* repositories, a wiki, issue tracker and more all in a single package that's easy to install (take a look at this month's step-by-step tutorial on page 66 for details of how to get up and running).

There are versions for any distro based on Debian, Ubuntu or Red Hat (unfortunately SUSE users seem to be neglected). If you want to keep your energy bills to a minimum, there's also a version for the Raspberry Pi, which is perfect for running at home, though we haven't tried this one so we can't comment on what the performance is like.

The web interface is one of the nicest we've seen for project hosting. It's all very straightforward, and most users should be able to use it without difficulty. Since it's all based on *Git*, the workflow will be familiar to anyone who's used any *Git*-based system in the past.

If you'd rather someone else did the hosting for you, but you still want to host your code on an open source platform, *GitLab* offers a series of options from free hosting (see **https://about.gitlab.com/gitlab-com** for details). There's also an enterprise level of *GitLab* which has more features than the fully open sourced community version (including Kerberos integration and better *Jira* support and improved user controls), however there's little there for open source projects.

Gitlab's web interface is one of the nicest we've seen for project hosting



The web interface to Gitlab is as good as any project hosting we've used, either open source or commercial.

PROJECT WEBSITE
https://gitlab.com

Web terminal

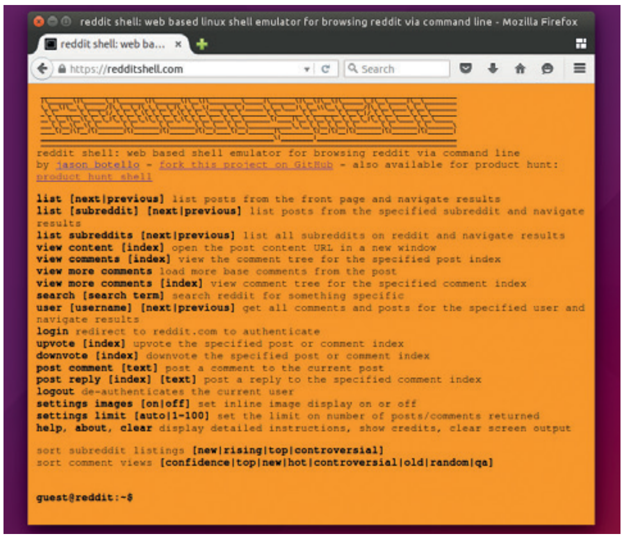
Reddit shell

For most of us, the CLI is nothing more than a useful tool to make powerful commands easy to use; for others it's the centrepiece of a cult. *Reddit Shell* is a tool for these CLI fanatics. It offers a terminal-style interface for the popular Reddit news aggregation site that enables you to use commands such as **ls** and **cd** to view and move through the various sections of the site (known as subreddits).

Reddit Shell runs inside a web browser rather than in a normal terminal, but the style of interaction is the same. The hosted version runs at **https://redditshell.com**, but the software is open source, so you can grab the code from **https://github.com/jasonbio/reddit-shell** and host it yourself if you're that way inclined (it's written in JavaScript so should be easy to run).

This isn't the first web-based terminal-style interface for a website. On April Fool's Day 2010, the geek web comic XKCD launched *UNIXKCD* (**http://uni.xkcd.com**) with a similar style interface.

We must admit, we can't think of a good reason to use either of these. In both cases, the ordinary website is easier and quicker to use, and if you really had an aversion to using the mouse, it would probably still be easier to use a browser extension that's designed for this rather than trying to find a terminal-style interface for every site you use. Perhaps, at a stretch, you could argue that these may make it a little easier to use these websites at work and hide the fact that you're not really working, but that's a bit of a stretch. This is all beside the point though – the reason that these exist isn't because of any need, but because they're good, clean fun for us terminal addicts.


reddit shell: web based linux shell emulator for browsing reddit via command line - Mozilla Firefox
reddit shell: web ba...
https://redditshell.com
Search

```
reddit shell: web based shell emulator for browsing reddit via command line
by jason botello - fork this project on GitHub - also available for product hunt:
product hunt shell

list [next|previous] list posts from the front page and navigate results
list [subreddit] [next|previous] list posts from the specified subreddit and navigate
results
list subreddits [next|previous] list all subreddits on reddit and navigate results
view content [index] open the post content URL in a new window
view comments [index] view the comment tree for the specified post index
view more comments load more base comments from the post
view more comments [index] view comment tree for the specified comment index
search [search term] search reddit for something specific
user [username] [next|previous] get all comments and posts for the specified user and
navigate results
login redirect to reddit.com to authenticate
upvote [index] upvote the specified post or comment index
downvote [index] downvote the specified post or comment index
post comment [text] post a comment to the current post
post reply [index] [text] post a reply to the specified comment index
logout de-authenticates the current user
settings images [on|off] set inline image display on or off
settings limit [auto|1-100] set the limit on number of posts/comments returned
help, about, clear display detailed instructions, show credits, clear screen output

sort subreddit listings [new|rising|top|controversial]
sort comment views [confidence|top|new|hot|controversial|old|random|qa]


guest@reddit:~$
```

Reddit Shell leaves us one step closer to getting rid of this pesky mouse.

PROJECT WEBSITE
https://redditshell.com

Programming language

FreeBASIC

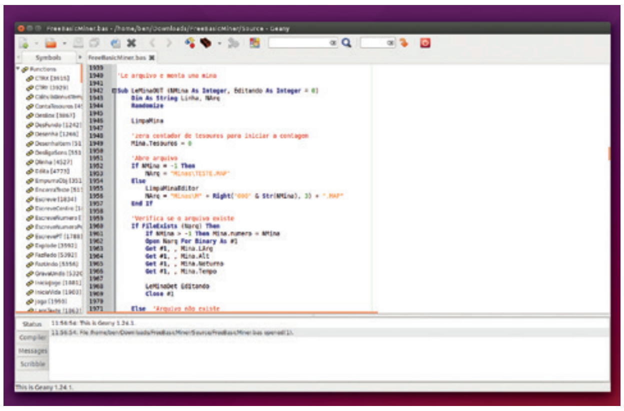
Beginner's All-purpose Symbolic Instruction Code (BASIC) came out in 1964 as a product of Dartmouth University. It was designed to allow students who didn't specialise in a computing-related field to write software. For the next 30 years, it was the dominant programming language for people new to computing. Whole generations of programmers first learned their craft on dialects of this language: AmigaBASIC, Commodore BASIC, BBCBASIC and QuickBASIC were some of the most popular.

Almost all of us here at Linux Voice first programmed in one of these, and we have fond memories of the time. *FreeBASIC* continues this legacy. It's a BASIC compiler that loosely follows the QuickBASIC version of the language. Unlike many versions of the language, *FreeBASIC* is just a compiler, so you're free to use it with any text editor or IDE.

BASIC may have grown up as the language for beginners, but this doesn't mean that you can't write serious software in it. If you don't believe us, just take a look at the FreeBASIC Games Directory (**http://games.freebasic.net**). Some are simple, some are half-finished and some are Windows-only (thanks to the QuickBASIC legacy) but many are complex games that are well written and run smoothly.

Cosmic poetry

There's a strange beauty in BASIC's language that, when written well, is somewhere between engineering and poetry. Other languages can have elegant code, but few have the majesty of well written BASIC with it's REMs (comments) and **For X=1 to 3 … Next** loops. Curly braces and more compressed grammar may make writing in other languages quicker, but it takes away a little of the fundamental joy of programming that remains in BASIC. Even though we rarely program in this language any more, it's nice to know that it's out there, still active and waiting for us should we ever wish to return to the fold.


```
'Le arquivo e monta uma mina

Sub LeMinaOUT (NMina As Integer, Editando As Integer = 0)
    Dim As String Linha, NArq
    Randomize

    LimpaMina

    'zera contador de tesouros para iniciar a contagem
    Mina.Tesouros = 0

    'Abre arquivo
    If NMina = -1 Then
        NArq = "Minas\TESTE.MAP"
    Else
        LimpaMinaEditor
        NArq = "Minas\M" + Right("000" & Str(NMina), 3) + ".MAP"
    End If

    'Verifica se o arquivo existe
    If FileExists (Narq) Then
        If NMina > -1 Then Mina.numero = NMina
        Open Narq For Binary As #1
        Get #1, , Mina.LArg
        Get #1, , Mina.Alt
        Get #1, , Mina.Noturno
        Get #1, , Mina.Tempo

        LeMinaDet Editando
        Close #1

    Else  'Arquivo não existe
```
```
11:56:54: This is Geany 1.24.1.
11:56:54: File /home/ben/Downloads/FreeBasicMiner/Source/FreeBasicMiner.bas opened(1).
```

Geany can provide a good development environment for programming in *FreeBASIC*.

PROJECT WEBSITE
www.freebasic.net

Distro builder

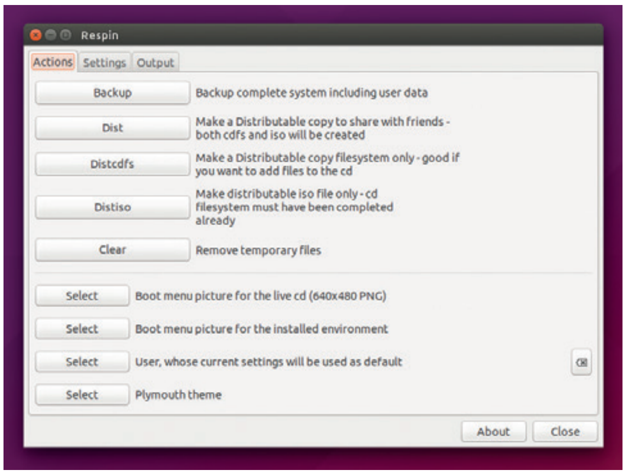
Linux Respin

Once upon a time there was a tool called *Remastersys* for building live and installable ISO files from Debian and Ubuntu systems. Unfortunately, distro development, like the tides, stops for no man, and new developments mean that the original tool no longer works. This left Debian and Ubuntu without an easy to use tool for customising their distros.

Fortunately, the story does not end there. The Copper Linux User Group in Arizona has taken up the mantle and continued working on the project under the new name of *Linux Respin*. While this name is fairly descriptive, it does make the tool almost impossible to find via a web search. Even when you have found the project's web page, the task doesn't get much easier. At least, when we tried, the download link was broken and there was no link to the original repository. We can save you from the ordeal and point you directly to the project's repository on GitHub, where you'll find Deb files for both Ubuntu and Debian: **https://github.com/chamuco/respin**.

Linux Respin comes in both command line and graphical versions, and it enables you to clone your system either to a full backup or only including the programs and not the data. Both versions are incredibility easy to use, and you can have your live system ready in just a few clicks. There are loads of uses for these customised live distros: creating a custom install CD specifically for an organisation, create a backup of your current setup, or even build your own distro.

Linux Respin isn't the only continuation of *Remastersys*. The Pinguy Linux team have created *Pinguy Builder* in a similar manner. At the moment, both tools seem very similar, with the only real differentiation being that *Pinguy Builder* doesn't support Debian.



Backup complete system including user data
Make a Distributable copy to share with friends - both cdfs and iso will be created
Make a Distributable copy filesystem only - good if you want to add files to the cd
Make distributable iso file only - cd filesystem must have been completed already
Remove temporary files
Boot menu picture for the live cd (640x480 PNG)
Boot menu picture for the installed environment
User, whose current settings will be used as default
Plymouth theme

Linux Respin's GTK interface makes building a new distro as easy as clicking a button.

PROJECT WEBSITE
www.linuxrespin.org

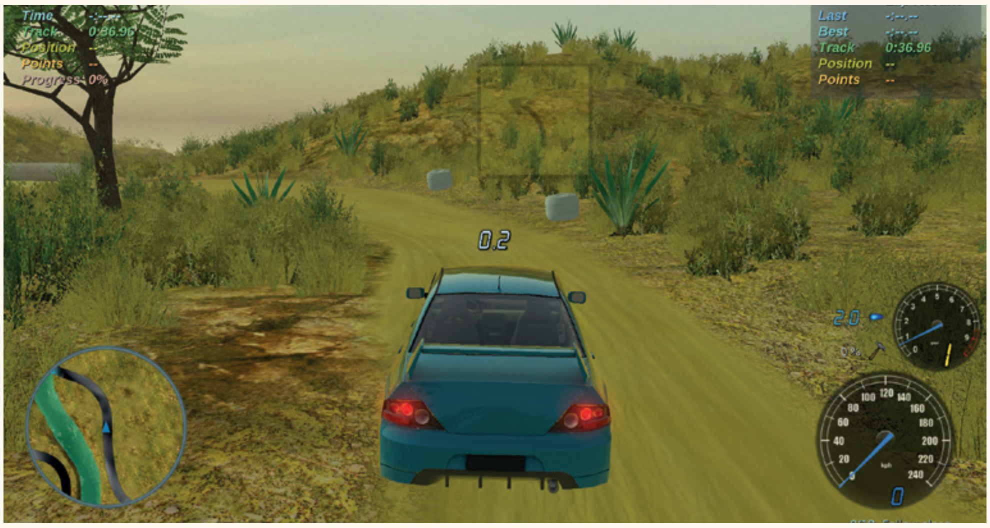
FOSSPICKS Brain Relaxers

Car racer

Stunt Rally

The problem with the real world is that it hurts too much if you skid off a road and hit a tree at 50 miles an hour. It might be fun to try, but the months of rehab and inflated car insurance means that it's just not worth it. Fortunately, thanks to computers, we can all experience the thrill of rushing round a corner in a safe, virtual world.

Stunt Rally gives us just such a world to race around in a wholly inappropriate manner. It's all about driving quickly round tracks with insufficient traction. In the latest version (2.6), there are 172 tracks in a wide variety of settings from jungles to deserts to alien planets. You can race around these using cars, motorbike and even *Star Wars*-esque land speeders.



All the fun of reckless driving without the pain, cost or legal trouble.

Stunt Rally features impressive graphics, even on our lowly test machine. You can adjust the level of detail to let the game to run well on whatever hardware you have.

If you fancy turning your hand to game design, *Stunt Rally* comes with a level designer, so you can race around whatever you can imagine, and even contribute back to help the game grow and let other people enjoy the fruits of your creation.

PROJECT WEBSITE
http://stuntrally.tuxfamily.org

Real-time strategy

OpenDungeons

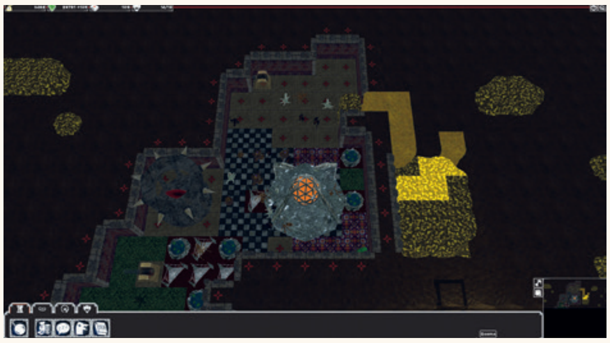
OpenDungeons is a real time strategy game in which you command a legion of creatures that crawl, creep and fly around in darkened caverns underground.

As the game progresses, you can build libraries where workers can research new items that you can use to expand and defend your dungeon. With food and experience, your workers will grow and develop into honed fighting machines. Using your dungeon as a base, you can then launch attacks on other dungeon masters, with the aim of achieving dungeon supremacy. This can be against real players in a networked game, or AI players.

The documentation is a little hidden, but can be found at **https://github.com/OpenDungeons/OpenDungeons/wiki**. It can take a little while to get used to some of the nuances, so it's worth having a quick scan through once you've got a feel for the game. For example, you have to explicitly set workers to task in the libraries and workshops if you want to reap the fruits of these rooms, and this won't happen automatically.

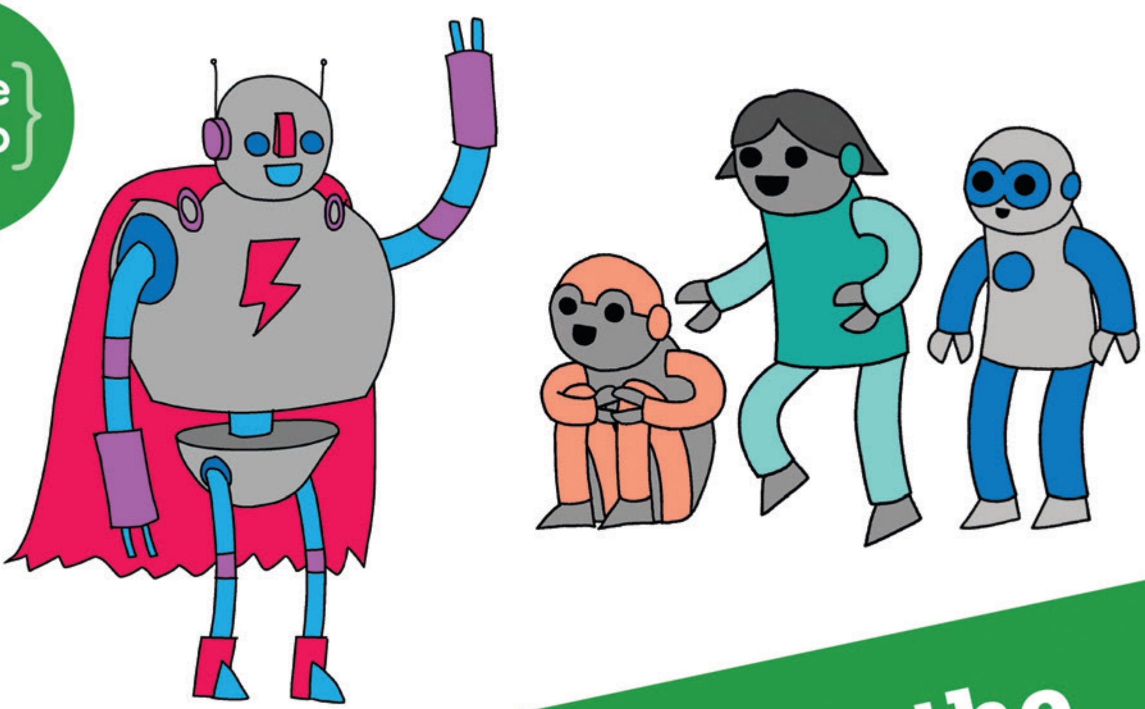
The graphics are better than most games of this ilk, and help make *OpenDungeon* a good, fun game to play once you've mastered the controls. The gameplay is a little less frantic than some other games in this genre, and there aren't many maps available, so it is a little short on content if you plan on mostly playing against the computer, but when playing against real people, there should be enough variation to keep you entertained (and avoiding work) for quite some time.

OpenDungeon is a good, fun game to play once you've mastered the controls


Take on other dark lords and build the ultimate dungeon.

PROJECT WEBSITE
https://opendungeons.github.io

{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 embarking on a crusade to get open source developers to document their work.

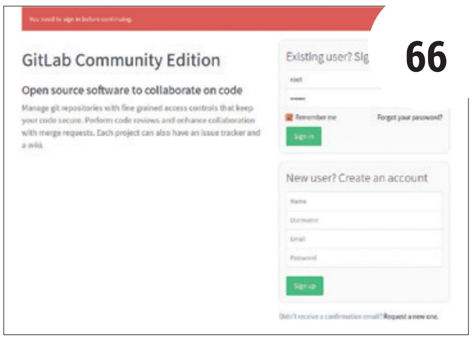
Programming has changed since I was a lad. Back in those days, if you wanted to do something, you had to write the code to do it. Nowadays, it seems, all you ever need to do is link a couple of libraries together and you can do anything.

The same thing, it seems, is happening with electronics. Recently I wanted to turn a broken 1940s radio into a Bluetooth speaker. All it needed was a Bluetooth audio module, amplifier and power supply. Wire them all together and it works. At least, that's the theory.

The problem with libraries and electronics modules is documentation. There are few things more frustrating than trying to work out how a poorly documented function works in a library, or what on earth the pin labelled 'EN' does on an electronics module. It's starting to feel like everything I need is already created, if only I could work out how to use it.

Geeks of the world, I urge you, step away from the IDE, put down your soldering iron, and take a little time to properly document your creations.
ben@linuxvoice.com

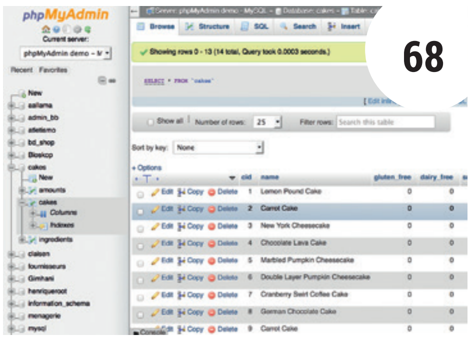
In this issue . . .


You need to sign in before continuing.
GitLab Community Edition
Open source software to collaborate on code
Existing user? Sig
root
Remember me
Forgot your password?
Sign in
New user? Create an account
Name
Username
Email
Password
Sign up
Didn't receive a confirmation email? Request a new one.
66

Manage git repositories with fine grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Host your own software projects with GitLab

Ditch commercial hosting providers and their proprietary software, and follow **Ben Everard**'s guide to open source hosting using *GitLab*.



Brush up your database skills with MariaDB

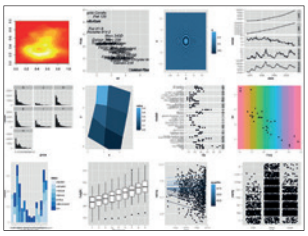
Databases can be tricky, but fear not: **Mike Saunders** is here to show you how to set up and use *MariaDB* as a web app back-end.

Raspberry Pi info centre 72
Les Pounder uses a Raspberry Pi and a Displayotron HAT to keep himself up to date.

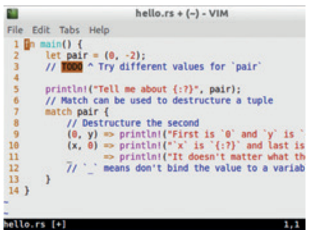
Vulnerability scanning 76
Set a robotic penetration tester to work. **Ben Everard** automates finding vulnerabilities.

Head tracking 80
Improve your gaming experience by turning yourself into a cyborg, like **Graham Morrison**.

Coding

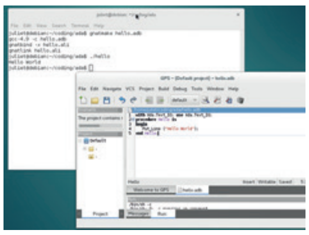


GNU R and Git 84
With GNU R , Git and a bit of statistical magic, **Robin Gower** assesses popular FOSS projects.



```
fn main() {
    let pair = (0, -2);
    // TODO ^ Try different values for `pair`

    println!("Tell me about {:?}", pair);
    // Match can be used to destructure a tuple
    match pair {
        // Destructure the second
        (0, y) => println!("First is `0` and `y` is `
        (x, 0) => println!("`x` is `{:?}` and last is
               => println!("It doesn't matter what th
        // `_` means don't bind the value to a variab
    }
}
hello.rs [+]                                    1,1
```

Get started with Rust 88
Mike Saunders investigates the new language from Mozilla that aims to make the web a bit safer.



Olde Code: Ada 90
Juliet Kemp's guide to computer history reaches the 70s with the Ada programming language.

HOST YOUR OWN SOURCE CODE WITH GITLAB

Run your open source project on your own open source server.

BEN EVERARD

WHY DO THIS?
- Keep your source code on servers you control
- Save on bandwidth by hosting locally
- Avoid using closed source software in your workflow

Since its creation as a source code management tool built specifically for the Linux kernel, *Git* has become the most popular option for hosting open source projects. This is partly fuelled by several web-based hosting providers that offer free hosting to open source projects. However, there are times when a public hosting system isn't what you need. Hosting your own code repository can be as simple as running a git server and giving everyone commandl-ine access. This works, but having a web interface to manage everything makes it all a bit nicer. *GitLab* provides this web interface along with a wiki for documentation and a bug tracker to keep an eye on any problems. It's a fully featured open source code hosting environment that you can set up and run on your own hardware. Let's get it up and running!

STEP BY STEP: INSTALL GITLAB

1 Get the dependencies

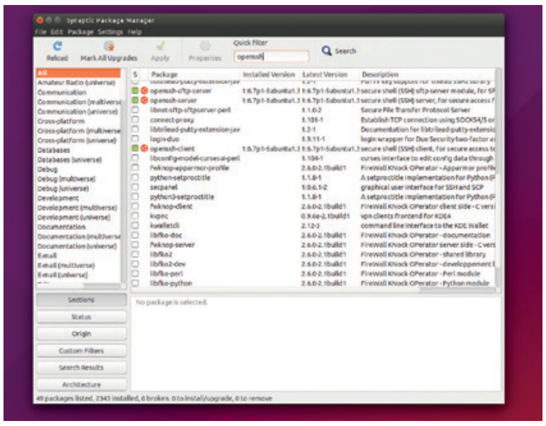
Before we get into *GitLab* properly, you need to get all the dependencies. The key ones are the *Curl* library, the OpenSSH server, an SMTP (email) server such as *Postfix* and the certificate authority certificates (if using a Debian-based system). In a Debian-based system, you can grab these with:

```
sudo apt-get install curl openssh-server ca-certificates
postfix
```

Alternatively, you can install them through a package manager GUI such as *Synaptic*.

If using Red Hat (or a clone), you'll also have to enable the SSH and *Postfix* services using *systemctrl* (for *Systemd* versions) or *service* (for older systems). Additionally, Red Hat users will need to open the HTTP port in their firewalls. See the *GitLab* documentation for more details if you're unsure of how to do this.

During this install, you may be asked what sort of profile you want for *Postfix*. In this case, select Internet Site.



```
S  Package                        Installed Version     Latest Version        Description
   openssh-sftp-server            1:6.7p1-5ubuntu1.3    1:6.7p1-5ubuntu1.3    secure shell (SSH) sftp server module, for SF
   openssh-server                 1:6.7p1-5ubuntu1.3    1:6.7p1-5ubuntu1.3    secure shell (SSH) server, for secure access f
   libnet-sftp-sftpserver-perl                          1.1.0-2               Secure File Transfer Protocol Server
   connect-proxy                                        1.105-1               Establish TCP connection using SOCKS4/5 or
   libtrilead-putty-extension-jav                       1.2-1                 Documentation for libtrilead-putty-extensio
   login-duo                                            1.9.11-1              login wrapper for Duo Security two-factor a
   openssh-client                 1:6.7p1-5ubuntu1.3    1:6.7p1-5ubuntu1.3    secure shell (SSH) client, for secure access to
   libconfig-model-cursesui-perl                        1.104-1               curses interface to edit config data through
   fwknop-apparmor-profile                              2.6.0-2.1build1       FireWall KNock OPerator - Apparmor profile
   python-setproctitle                                  1.1.8-1               A setproctitle implementation for Python (P
   secpanel                                             1:0.6.1-2             graphical user interface for SSH and SCP
   python3-setproctitle                                 1.1.8-1               A setproctitle implementation for Python (P
   fwknop-client                                        2.6.0-2.1build1       FireWall KNock OPerator client side - C versi
   kvpnc                                                0.9.6a-2.1build1      vpn-clients frontend for KDE4
   kwalletcli                                           2.12-3                command line interface to the KDE wallet
   libfko-doc                                           2.6.0-2.1build1       FireWall KNock OPerator - documentation
   fwknop-server                                        2.6.0-2.1build1       FireWall KNock OPerator server side - C vers
   libfko2                                              2.6.0-2.1build1       FireWall KNock OPerator - shared library
   libfko2-dev                                          2.6.0-2.1build1       FireWall KNock OPerator - developpement l
   libfko-perl                                          2.6.0-2.1build1       FireWall KNock OPerator - Perl module
   libfko-python                                        2.6.0-2.1build1       FireWall KNock OPerator - Python module

No package is selected.

49 packages listed, 2343 installed, 0 broken. 0 to install/upgrade, 0 to remove
```

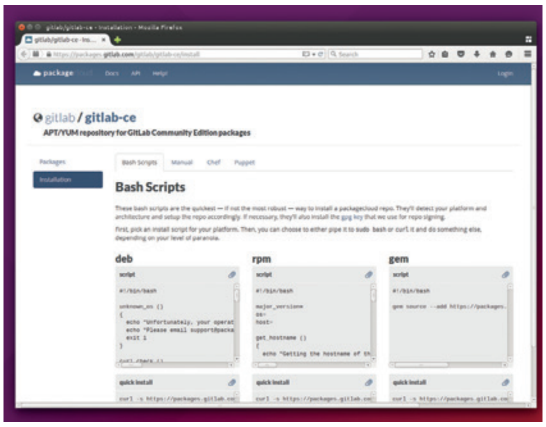
2 Get GitLab

With the dependencies installed, we can now get the main software. The *GitLab* team have put together a script that will detect which distro you're running and add the appropriate repositories. You can download and run this in one go with:

```
curl https://packages.gitlab.com/install/repositories/
gitlab/gitlab-ce/script.deb.sh | sudo bash
```

If you're using a Red Hat-based distro, you'll need to change **script.deb.sh** to **script.rpm.sh**.

If you'd rather not pipe a script straight from the internet into **sudo bash**, you can perform the process manually by following the steps here **https://packages.gitlab.com/gitlab/gitlab-ce/install**. Once the script has finished running, you can install the **gitlab-ce** package. This script doesn't work in Ubuntu 15.04, so you'll need to edit the file **/etc/apt/sources.list.d/gitlab_gitlab-ce.list** and change any occurrences of **vivid** to **trusty**. After this, run **sudo apt-get update** and you'll be able to install **gitlab-ce**.


gitlab/gitlab-ce - Installation - Mozilla Firefox
gitlab/gitlab-ce - Ins...
https://packages.gitlab.com/gitlab/gitlab-ce/install
Search
package
Docs
API
Help!
Login
gitlab / gitlab-ce
Packages
Installation
Bash Scripts
Manual
Chef
Puppet
deb
rpm
gem
script
quick install

APT/YUM repository for GitLab Community Edition packages

Bash Scripts

These bash scripts are the quickest — if not the most robust — way to install a packagecloud repo. They'll detect your platform and architecture and setup the repo accordingly. If necessary, they'll also install the gpg key that we use for repo signing.

First, pick an install script for your platform. Then, you can choose to either pipe it to sudo bash or curl it and do something else, depending on your level of paranoia.

```
#!/bin/bash

unknown_os ()
{
  echo "Unfortunately, your operat
  echo "Please email support@packa
  exit 1
}
```

```
#!/bin/bash

major_version=
os=
host=

get_hostname ()
{
  echo "Getting the hostname of th
```

```
#!/bin/bash

gem source --add https://packages.
```

```
curl -s https://packages.gitlab.co
```

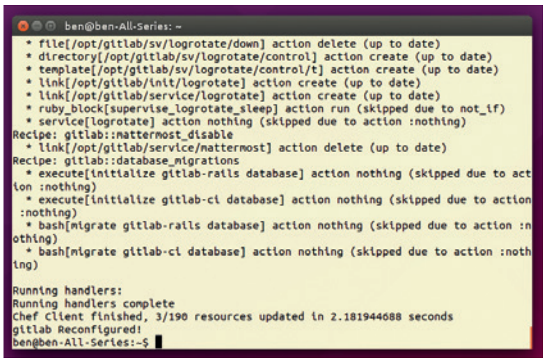
3 Reconfigure

By this point, all the *GitLab* software is installed, but it's not yet configured or running. Fortunately, *GitLab* comes with a tool to make this all really straightforward. You just need to run the following:

```
sudo gitlab-ctl reconfigure
```

The **gitlab-ctl** program can be used to manage the *GitLab* server. You can see all the possible options by passing it the **--help** flag, but a few of the most useful are: **restart**, **status** and **stop**. These are all pretty self-explanatory.

If you end up with a borked install, you can try passing the **cleanse** option, which will delete all the data and leave you with a clean version (make sure you've got backups of any important data before running this). If you're trying to debug a problem, the **tail** option will output the logs of all the services, so you can see a bit more about what's going on.



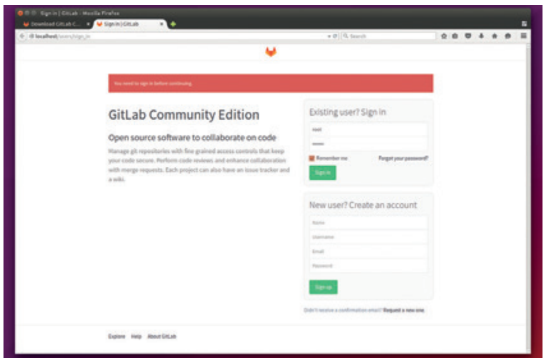
```
ben@ben-All-Series: ~
  * file[/opt/gitlab/sv/logrotate/down] action delete (up to date)
  * directory[/opt/gitlab/sv/logrotate/control] action create (up to date)
  * template[/opt/gitlab/sv/logrotate/control/t] action create (up to date)
  * link[/opt/gitlab/init/logrotate] action create (up to date)
  * link[/opt/gitlab/service/logrotate] action create (up to date)
  * ruby_block[supervise_logrotate_sleep] action run (skipped due to not_if)
  * service[logrotate] action nothing (skipped due to action :nothing)
Recipe: gitlab::mattermost_disable
  * link[/opt/gitlab/service/mattermost] action delete (up to date)
Recipe: gitlab::database_migrations
  * execute[initialize gitlab-rails database] action nothing (skipped due to act
ion :nothing)
  * execute[initialize gitlab-ci database] action nothing (skipped due to action
 :nothing)
  * bash[migrate gitlab-rails database] action nothing (skipped due to action :n
othing)
  * bash[migrate gitlab-ci database] action nothing (skipped due to action :noth
ing)

Running handlers:
Running handlers complete
Chef Client finished, 3/190 resources updated in 2.181944688 seconds
gitlab Reconfigured!
ben@ben-All-Series:~$
```

4 Login

That's all you need to do! *GitLab* is now installed and running on port 80 of the machine. Just point your browser to **http://localhost** and you should see the login prompt. The default username and password is root/5iveL!fe, however when you log in, you'll be prompted to immediately change this password.

After logging in a second time (with the new password), you'll be taken to the admin panel. Here you can see the status of *GitLab* . At this point, it'll be empty, but this will be a useful place to see what's happening. You can also keep an eye on updates here.

You can get back to this admin page from other pages in *GitLab* by clicking on the spanner icon in the top right-hand corner of the screen. Using this panel you can also set other options like alternative authentication including OAuth to integrate *GitLab* with other systems.

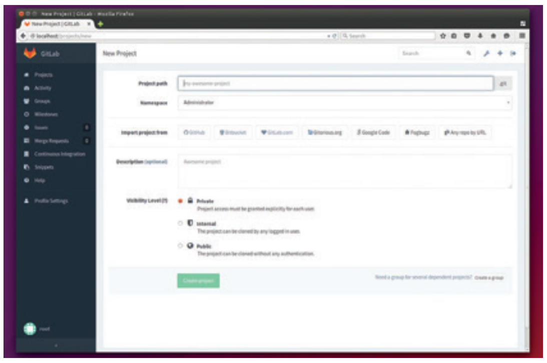

GitLab Community Edition
Open source software to collaborate on code
Existing user? Sign in
New user? Create an account

5 Create project

The main reason for going to all this trouble is to host projects, so let's create one! Click in the plus icon in the top right-hand corner of the screen and you'll see the New Project page. You can create a blank project, or import one from an external *Git* repository.

Creating a project creates a *Git* store for the source code. It also provides a wiki for documentation that defaults to markdown formatting, an issue tracker to help you fix bugs, milestone tracking and a few other useful project management tools. All in all, it gives you most features you'll need to keep your project ticking along smoothly.

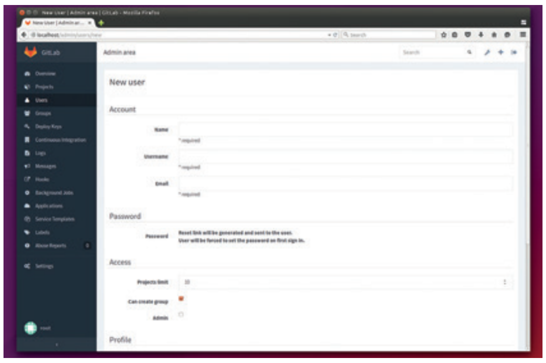
The first step in a new project is usually creating the **readme.md** file to let everyone know what's going on with it. Click on the link on the main page to create this, and your project will be ready for contributions.



6 Create users

Now you've got your server running and created your new project, the only thing left to do is to add users. As an administrator, you can create new users by going to the admin area and clicking on New Users. This is fine if you want a few users, but it's not a great idea if you need to add a lot. A better option is to allow users to register themselves. Go to the Settings menu in the admin panel, and scroll down to Sign-In Restrictions. Here you can set the options whereby users register themselves.

The best settings depend entirely on your use-case. You may wish to open the system up to anyone who wants an account, or you can control it more tightly – the best way to restrict users is by the domain of their email. You can also configure OAuth if you want to enable restrictions like this. LV



SERVER 101: BRUSH UP YOUR DATABASE SKILLS

Part 1: Most major websites (and apps) are built on databases. Here's why they're important, how they work, and how to set one up from scratch.

MIKE SAUNDERS

WHY DO THIS?

- Discover how web apps work under the hood
- Explore data stored by WordPress, OwnCloud and co.
- Learn SQL to perform powerful search queries

When compared with desktop environments, distros, web browsers, games and other fast-moving end-user software, databases may seem like the most tedious of topics that only beard-endowed geeks get excited about. But databases are fascinating and well worth learning, even if you never use one directly.

Without databases, most major websites would be incredibly slow, inefficient and prone to major problems – and plenty of desktop apps use databases as well. If you have a lot of information to store in a reliable and easily searchable fashion, even if it's your own personal beer mat collection, using a database is a good idea.

But what exactly is a database? Imagine that they didn't exist, and every program and website had to store data in its own format. Some software would write data to text files, perhaps in CSV or XML format. Other programs would use proprietary binary formats. Every piece of software would have its own data storage, loading and searching routines, duplicating a load of effort and making it extremely difficult for other programs to share data.

Databases fix this by handling all the dirty work of storing, loading and searching data. A program that uses the database – be it a website or a desktop application – speaks to the database in a standardised language, so that other programs can query the same database and extract or upload information from it. The database worries about data integrity, backups, duplicated entries and so forth, so the programs that use it can focus on their own features instead.

Without databases, most major websites would be incredibly slow, inefficient and prone to major problems

How it all works

Most databases store information in tables, which look somewhat like spreadsheets. A table consists of one or more columns, which define the types of data included in a table, along with rows, which represent the entries. Look at this example:

MariaDB has this peppy looking seal as its mascot, although it doesn't appear to have a name. Any suggestions, readers?

ID	Name	Login
(INTEGER)	(VARCHAR(20))	(Date)
1	Mike	2015-01-12
2	Ben	2015-04-25
3	Graham	2015-10-02

This table stores information on user logins for a server, and has three columns: a unique ID for the login event, the username, and the date on which it took place (year-month-day).

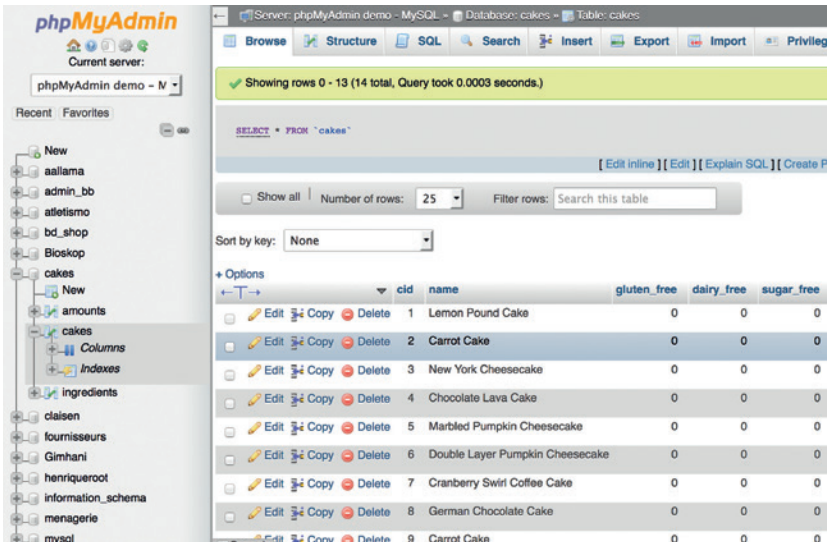
Databases can store all manner of information, but we can make them more reliable and easier to search by enforcing the type of data that a column can contain (a bit like static typing in programming). So we say here that the ID column can only contain integer numbers, while the Name column can contain varying characters (ie a text string) up to a maximum of 20 characters, while the Login column can only contain a date.

A single database can contain many different tables, and when a program (P) speaks to a database server (D) to update data in a table (T), the

Choices, choices

MariaDB (and formerly *MySQL*) is the most common database for low-to-mid-end tasks, such as blogs, simple server applications and moderately popular websites. If you need to chew through a lot of data, however, a good alternative is *PostgreSQL* (**www.postgresql.org**). While its name is rather a mouthful, *PostgreSQL* has enjoyed a reputation for having advanced features way beyond *MariaDB*, at the expense of some performance and ease-of-use. In recent years, however, both databases have come closer in terms of feature sets and speed, and they're providing healthy competition for each other.

If you're writing some desktop software and want the benefits of a database without having to run one as a separate server, consider *SQLite* (**www.sqlite.org**). This is a C library that lets you embed a database into a program to get its benefits – data integrity, powerful search facilities and so forth. As an example, the *Firefox* web browser uses *SQLite* to store configuration data, bookmarks and other information.



Showing rows 0 - 13 (14 total, Query took 0.0003 seconds.)

```
SELECT * FROM `cakes`
```

cid	name	gluten_free	dairy_free	sugar_free
1	Lemon Pound Cake	0	0	0
2	Carrot Cake	0	0	0
3	New York Cheesecake	0	0	0
4	Chocolate Lava Cake	0	0	0
5	Marbled Pumpkin Cheesecake	0	0	0
6	Double Layer Pumpkin Cheesecake	0	0	0
7	Cranberry Swirl Coffee Cake	0	0	0
8	German Chocolate Cake	0	0	0
9	Carrot Cake	0	0	0

In this tutorial we're interacting with *MariaDB* via its command line tool, but web-based software to manage databases, such as *phpMyAdmin*, is available as well.

conversation – usually over a network socket – goes like this:
P: Hi, I'm Program P. Can I have access to the login database please?
D: One sec... Let me check that you're allowed to access that particular database. Ah yes, you can – but what's your password?
P: It's <password>.
D: OK, you're in. What do you want to do?
P: Can you insert this data into table T please: "Mike, 2015-01-12"
D: Done – that entry was added and has the ID 5.
P: Thanks! I'm off now.

Note here that the program only supplied data for the Name and Login columns – but not for the ID. That's because the database generates (and increments) the ID itself, so that there are no duplicates, and in this way we can turn the ID into a primary key. What's that? Well, it's a unique reference to a particular entry in the database, so we can have multiple entries with the same data (eg if Graham logged in many times on the same day) but with their own IDs.

And this is what makes relational databases work. With unique primary keys, we can create other tables that contain additional data, for example:

ID	Command	Exit code
1	df -h	0
2	crontab -e	1
3	shutdown	1

This table supplements the login one with extra information, and shares the same primary key in the ID column. So from the first table we know that Ben's login on 2015-04-25 has the ID number 2, and we can then use that ID in this table to see what command he issued (**crontab -e**).

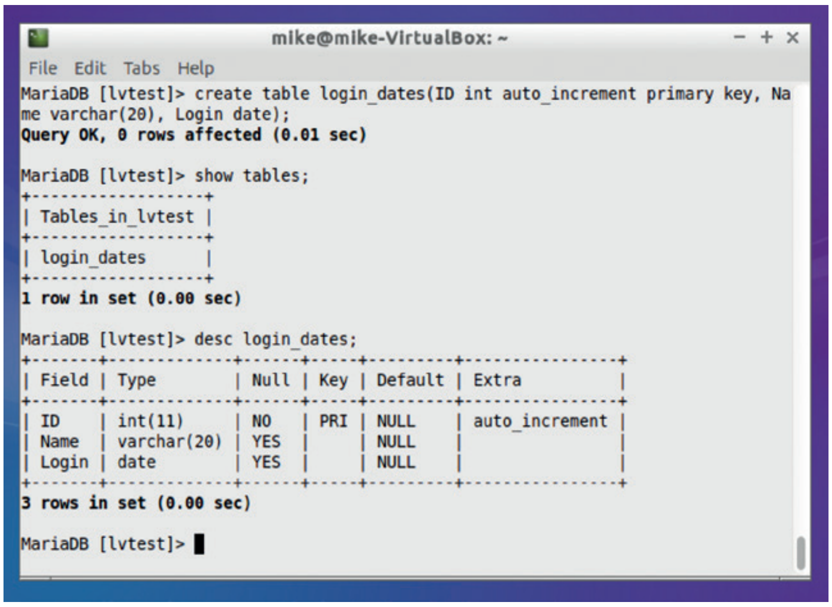
This lets us create lots of tables containing different data, with all the entries linked together by the same ID. You could, of course, put everything together in a single table, but this is a more efficient and secure approach. It's possible to restrict programs that access the database to specific tables – so in a HR system, for example, the kitchen staff are able to access an employee's dietary preferences table in the database without seeing their salary information and other personal details. It also gives you more flexibility with backups, in that you can make regular (daily) backups of the most important tables to save time, and less regular (weekly) backups of the whole database which may contain a lot of non-essential cruft.

A modification operation on a table – be it inserting a new entry, modifying an existing entry or deleting one – is known as a transaction, and good database software adheres to ACID principles. The letters mean:

- **Atomicity** A transaction should either work completely, or not at all. You won't want an entry containing a mixture of old and new data because something failed during a modification.
- **Consistency** The data must match the constraints of the database. For instance, there cannot be multiple duplicate primary key entries, and if a column specifies that it should only store integer numbers, a transaction which tries to insert text into that column will fail.
- **Isolation** If multiple programs are connected to the same database and are trying to modify the same table simultaneously, the database software should handle this in an orderly fashion. Each transaction should be handled in isolation from the others, so that half-finished transactions don't get overwritten by others.

PRO TIP

Made a mistake creating a table? You can delete it using the **drop** command. For instance, if you entered a typo when creating **login_dates**, you can remove it straight away by entering **drop table login_dates;**. Be careful with this command though – it doesn't ask you if you're sure before deleting the table and its data!



```
MariaDB [lvtest]> create table login_dates(ID int auto_increment primary key, Na
me varchar(20), Login date);
Query OK, 0 rows affected (0.01 sec)

MariaDB [lvtest]> show tables;
+------------------+
| Tables_in_lvtest |
+------------------+
| login_dates      |
+------------------+
1 row in set (0.00 sec)

MariaDB [lvtest]> desc login_dates;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| ID    | int(11)     | NO   | PRI | NULL    | auto_increment |
| Name  | varchar(20) | YES  |     | NULL    |                |
| Login | date        | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

MariaDB [lvtest]> 
```

Screenshot 1: Here we create a table, specifying its column names and the type of data they should contain. Then we view a description of it.

- **Durability** Once a transaction is done, the data should be stored permanently – even if the OS crashes or there's a power loss.

The isolation aspect is especially important when it comes to large-scale websites. Imagine a busy forum written in PHP with thousands of users online simultaneously: you end up with thousands of PHP processes prodding the forum database at the same time, all trying to update tables as users write and edit new posts, and the database needs to handle this sanely.

> We're going to use MariaDB, a popular database that was forked from MySQL after the latter was acquired by Oracle

Let's get practical

So, that's enough theory – let's put it into practice by creating a database ourselves. If you've ever installed *WordPress*, *OwnCloud* or similar software that uses a database, you may be familiar with part of this process. Many web apps automate the job of creating a database, which is a handy time-saver, but it shields you from all the goings-on under the hood. Some software requires that you create a database by hand – so it's well worth learning the skills.

For this we're going to use *MariaDB* (**www.mariadb.org**), a very popular open source database that was forked off from *MySQL* after the latter was acquired by Oracle and concerns were raised about its future. *MariaDB* is used by many well-known web apps and services such as the aforementioned *WordPress* and *OwnCloud*, and it's fairly easy to get started with. Most distros have it in their package repositories – so install it using your package manager or via the command line. For instance, in Ubuntu-based distros:

```
sudo apt-get install mariadb-server
```

This downloads and installs the database server, and starts it running as a background service (**mysqld**). Even though *MariaDB* is a fork of *MySQL*, it still uses the same commands and executable names as the old project, making it a drop-in replacement. The first thing you'll want to do is to secure the *MariaDB* installation as follows:

```
sudo mysql_secure_installation
```

MariaDB has its own "root" administrator account (separate from root on the operating system) which is used to add new users and perform various administrative tasks. *MariaDB* will ask you for the current root password, which is blank after a fresh installation, so just hit Enter. It will then ask you to create a new password for root, so type something suitable here.

Clean up your accounts

Next up, the installation script will offer to remove the anonymous testing user account – definitely a good idea – and disallow root logins over the network. Follow the remaining steps by entering Y, and then you'll be dropped back at the shell prompt. Now we want to create a normal (non-root) user account and a database, and set up permissions so that the new user can modify the database:

```
sudo mysql -uroot -p
```

This logs us into *MariaDB* as the root user with password authentication, so enter the password you provided during the secure installation phase. Then enter these commands:

```
create database lvtest;
create user 'lvuser'@'localhost' identified by 'pass123';
grant all privileges on lvtest.* to 'lvuser'@'localhost';
```

These steps are largely self-explanatory: we create a new database called **lvtest**, create a user called **lvuser** on the local machine with the password 'pass123', and then give this user all privileges (read, write, delete etc) for the **lvtest** database. In the **lvtest.*** part, the asterisk is a wildcard and refers to every table. You could replace the asterisk with the name of a specific table if you wanted to restrict access.

Speaking the same language

The commands that we use here to interact with the database are from SQL, the Structured Query Language. Not only does this provide a human-readable way to work with data, but it's also standardised and lets you use the same commands across different database programs. This is a huge benefit if you need to switch to more powerful database software – for instance, if you've built a website that is growing rapidly and your current database is crumbling under the load.

Many database programs also provide multiple options for storing data on disk. These "storage engines" vary in the way they structure and save data onto the disk, so some are optimised for maximum performance, whereas others are designed with extreme reliability in mind. It's not something you generally need to concern yourself, but if you're interested in learning more, see **https://mariadb.com/kb/en/mariadb/storage-engines**.

But we don't have any tables yet, so enter **exit** (or hit Ctrl+D like in a terminal window) to log out of the *MariaDB* root account, and then log in with the new account we created like so:

```
mysql -ulvuser -p
```

Enter **pass123** as the password and you'll land at the *MariaDB* prompt. Let's start building a database! We'll create the example login information tables that we used early on in the tutorial. First up we need to select the database to work with:

```
use lvtest;
```

Now we create a table, specifying the columns and types of data they should contain:

```
create table login_dates(ID int auto_increment primary
key, Name varchar(20), Login date);
```

When creating a table, we first specify the name of the column (ID, Name, Login as per earlier) followed by its data type (**int** for an integer number, **varchar(20)** for text up to 20 characters, and a **date**). We can also provide extra parameters for the column – like in the ID column, where we say that the number should go up by one each time a row is added, and that it should also be the primary key.

If you now enter **show tables;** you'll see that **login_dates** is now provided in the **lvtest** database, and if you enter **desc login_dates** you'll be presented with a description of the table, showing its columns (aka fields) and the type of data that they can contain, as in screenshot 1. Now let's populate the table with data:

```
insert into login_dates values (0, 'Mike', '2015-01-12');
```

Repeat this command twice more with the data from the table earlier in the tutorial (Ben 2015-04-25 and Graham 2015-10-02), and note that we specify zero for the ID column. This forces the database to generate its own numbers starting from 1 and automatically incremented as described when we created the table. Once you've inputted the data, view it like so:

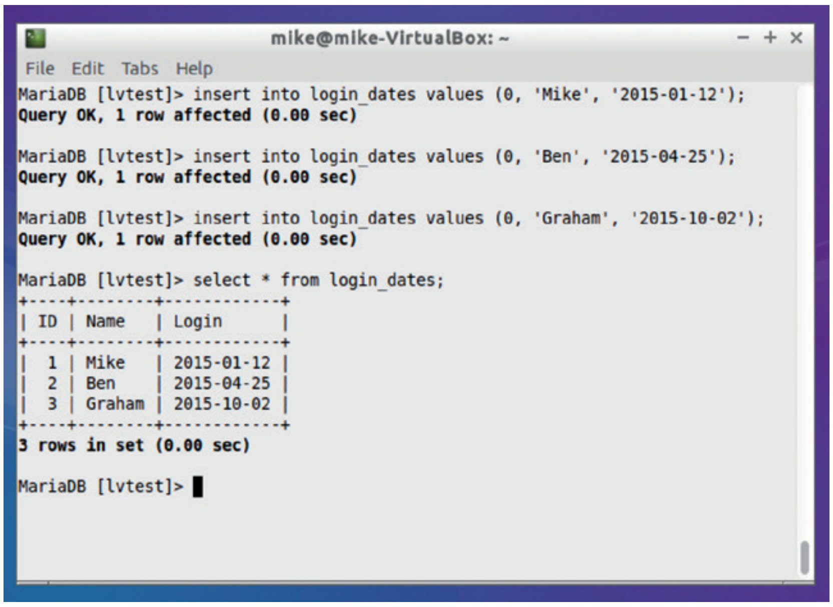
```
select * from login_dates;
```

This **select** command is mightily useful and lets you extract all kinds of information from a database. In this case it shows all data in a table (the asterisk being a wildcard again), as in screenshot 2. But why is the command called **select** and not something more meaningful like **show** or **view**? Well, in this case we're doing nothing especially important with the data – we're just looking at it. But in a typical real-world scenario, where a program retrieves data from a database, it will go on to do other things with the data: process it, combine it with something else, and so forth. So you ask the database to **select** the items you want according to your search terms, and then you perform an action on the selection.

It's also possible to narrow down the data that's retrieved from a **select** command, eg:

```
select * from login_dates where Name = 'Ben';
```

In this case, we're adding a restriction to the data that's returned. We're saying: select all data from the table **login_dates**, but only where 'Ben' appears in the name column. You can string together multiple



```
mike@mike-VirtualBox: ~
File  Edit  Tabs  Help
MariaDB [lvtest]> insert into login_dates values (0, 'Mike', '2015-01-12');
Query OK, 1 row affected (0.00 sec)

MariaDB [lvtest]> insert into login_dates values (0, 'Ben', '2015-04-25');
Query OK, 1 row affected (0.00 sec)

MariaDB [lvtest]> insert into login_dates values (0, 'Graham', '2015-10-02');
Query OK, 1 row affected (0.00 sec)

MariaDB [lvtest]> select * from login_dates;
+----+--------+------------+
| ID | Name   | Login      |
+----+--------+------------+
|  1 | Mike   | 2015-01-12 |
|  2 | Ben    | 2015-04-25 |
|  3 | Graham | 2015-10-02 |
+----+--------+------------+
3 rows in set (0.00 sec)

MariaDB [lvtest]> ▌
```

Screenshot 2: After carefully inputting the data into the table, we can view it using the **select** command.

where operations using **and** keywords, allowing you to perform more complicated searches. Indeed, some search queries are so long and intricate that they have more in common with a programming language than a simple query language...

Of course, it's possible to use other operators with searches as well. Take this for instance:

```
select * from login_dates where Name != 'Ben';
```

This shows the rows which don't contain 'Ben' in the Name column. You can perform searches using greater-than or less-than parameters as well:

```
select * from login_dates where ID > 2;
```

Note that text strings are placed inside single quotation marks, whereas numbers don't require them. Also note that the *MariaDB* command line client supports many time-saving features that are included in *Bash*, such as command history (use the arrow keys to cycle backwards through previously entered commands), and Ctrl+R followed by text to find the most recent command that included that text. Tab completion is also supported for table and column names.

PRO TIP

MariaDB commands and parameters are not case-sensitive, and many administrators use upper-case to distinguish them from actual data. So if you look at other command examples on the web, you may see them written like **CREATE DATABASE lvtest;**. It's purely a matter of taste, but if you're doing a lot of database work – especially in scripts – it can be useful for clarity.

Coming up next!

So there we have it: you now know the basics of how databases work, how to create them, and how to manage data inside them. The skills you're learning here are applicable to other database servers, and next issue we'll go further by looking at more advanced commands to manipulate and search through data and integrate with websites.

Mike Saunders is working on his own database, MikeSQL, written entirely in 16-bit x86 assembly language.

RASPBERRY PI: INFORMATION CENTRE

Les Pounder hacks together an information centre to provide the latest Morrissey news.

LES POUNDER

WHY DO THIS?
- Learn Python
- How to parse RSS feeds
- Create functions
- Use loops
- Create variables

TOOLS REQUIRED
- A Raspberry Pi 2 or A+ B+
- Displayotron HAT from Pimoroni
- Ethernet or Wi-Fi dongle
- 3.5mm speaker
- Power supply or USB battery

Information is all around us and it manifests itself in many different ways. But what remains constant is our need to consume and understand information, and in this project we shall use a Raspberry Pi and a Displayotron HAT from Pimoroni (**https://shop.pimoroni.com/products/displayotron-3000**) as an appliance that delivers news and information in two ways: firstly it will act as an interface for streaming internet radio, and secondly it will be an output device that shows the latest news from the BBC as a scrolling "ticker tape" on the LCD screen. We shall get the news via an RSS (Really Simple Syndication) feed and use a Python library to interpret the information and then present it on the LCD screen.

The Displayotron is easily fitted to all 40 of the extended GPIO pins found on the Raspberry Pi A+, B+ and Pi 2. Add-on boards should only be attached or removed while the power is off, otherwise you may damage either your Pi or the board. With the Displayotron fitted, connect the rest of your peripherals and boot the Raspberry Pi to the desktop. You'll need to ensure that your Raspberry Pi is also connected to the internet via an Ethernet or Wi-Fi dongle. Audio playback can also be configured using the new menu located in the top-right of the Raspbian desktop. Just right-click on the Volume icon and select the correct output. We used a small 3.5mm speaker for our project.

> Our device will show the latest news from the BBC as a scrolling ticker tape on the LCD screen

Set up the software

After booting to the desktop we will need to install the software that powers the project, and this comes in three stages.

We start by installing the Displayotron HAT library. With the terminal open enter the following command and press the Enter key to start.

```
curl get.pimoroni.com/dot3k | bash
```

This will run an installation script created by

The finished project can be easily powered from a USB battery giving a portable solution.

Pimoroni. You are safe to answer Yes to all of the installation questions. Once this is complete we shall test that it is working correctly. In the terminal type.

```
sudo idle &
```

With the *Idle* Python editor open we click on File > New to create a blank document. Save this as **LCDTest.py**.

```
import dothat.lcd as l
import dothat.backlight as b
b.rgb(255,0,0)
l.write("Hello World")
```

In the blank document we enter the code, which will import the LCD and backlight libraries as **l** and **b**. We then set the backlight colour to full red using the RGB values 255,0,0. Lastly we write the famous words "Hello World" to the LCD. Save the code and click on Run > Run Module to see the output on the Displayotron.

With a successful test we now move to installing the Python library that will handle RSS feeds; this is called *Feedparser*, and we can install it from the terminal as so.

```
sudo pip install feedparser
```

This uses the *pip* Python package manager to install and configure the *Feedparser* library.

Our last stage of software installation is the *VLC* library for Python, which can be downloaded via the terminal. Create a new directory called **Infocentre**

Taking your project anywhere

For this project we don't need a powerful computer (we've used a Raspberry Pi 2). Rather we would benefit from a lower power drain that would enable our project to become mobile. Step forward the Raspberry Pi A+, which has all of the GPIO pins and runs the same Raspbian operating system but uses the older BCM2835 system on a chip found on the older Raspberry Pis (up to the B+ in late 2014). Retailing at under £20 and very power efficient, the A+ is a great board for building embedded projects such as robotics, weather stations and our information centre project. Using the A+ with a USB dongle is possible, but as the A+ only has one USB port you'll need a powered USB hub in order to configure your dongle and connect a keyboard and mouse. It would also be prudent to set up SSH access, via the **raspi-config** terminal command, and fix the IP address of your Pi. Once configuration is complete you can plug the Wi-Fi dongle directly into your Pi and SSH in to the Pi to start your script. Or you could use **cron** to launch your script on boot. USB battery packs can be picked up for around £10 and should supply enough power for your project.

and ensure that you are inside that directory before downloading the file.

```
wget http://bit.ly/LV21-VLC -O vlc.py
```

This will download the file and rename it to **vlc.py**, which is required for this project. The **vlc.py** library needs to be in the same directory as our project code, as the library has not been installed system-wide; rather it is an external file that we are importing into our code.

So our installation is complete, now we start our project. In the terminal we will open the *Idle* Python editor again.

```
sudo idle &
```

With *Idle* open, click on File > New to create a new blank document and save it as **infocentre.py** before continuing.

```
import vlc, time
import dothat.touch as j
import dothat.lcd as l
import dothat.backlight as b
import signal
import feedparser
```

Our first section of code starts with a series of **import** commands; these bring in the various external libraries that we shall be using. Firstly we import the **vlc** library and the **time** library, which we use to control timings for various functions. From the **dothat** library, which is the Displayotron's library, we import the touch, LCD and backlight functions. For each of these imports we use a short reference, in this case **j** for touch, **l** for lcd and **b** for backlight. Next we import the signal library and lastly the **feedparser** library.

We now move on to creating a series of variables, our first of which is a global variable called **p** that will be used between functions that we shall create later. A global variable is needed as otherwise the variable would be "local": in other words it could only be used inside of the function and not between functions. The other variables are the names of radio stations that I like to listen to, we use their names to contain the link to the audio stream.

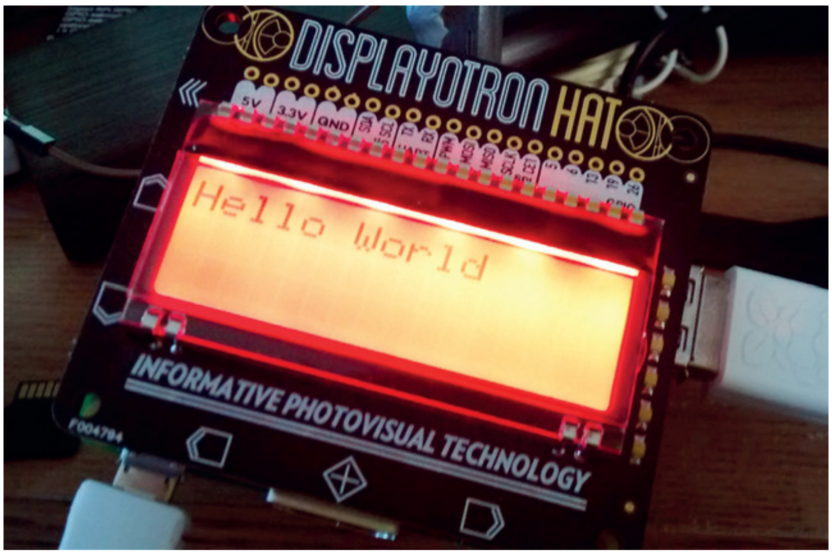
We now move to create a series of functions that will handle different aspects of the project. Our first function handles reading RSS feeds.

```
def feedme(feed):
    feed = feedparser.parse(feed.encode('utf-8'))
    for i in range(6):
        b.graph_set_led_state(i,1)
        print(feed['entries'][i]['title'])
        scrollText(feed['entries'][i]['title'])
    b.graph_off()
```

We start our function by giving it a name, in this case **feedme**. This feed will take an argument, which will be the web address for the RSS feed. The RSS feed passed to the function is then parsed using **feedparser** and encoded to utf-8, which sanitises the data so that we minimise any weird characters on the LCD.

We then save the results to the variable **feed**. Next we create a **for** loop which will repeat six times, giving us the top six headlines from the RSS feed. Why six? Well the Displayotron has six bar graph LEDs located to the right of the LCD screen, and we shall use those to indicate our progress through the feed. We do that by changing the LED state to 1, or on, for each LED in the column. With the LED lit we then print the RSS title to the shell for debug purposes and then call another function called *scrollText*, which we shall create later. Once the **for** loop has iterated six times, the loop will end and the bar graph LED will be turned off.

```
def scrollText(scrollBlurb):
  if len(scrollBlurb) > 16:
    padding = " " * 16
    scrollBlurb = padding + scrollBlurb + " "
    for i in range(0,len(scrollBlurb)):
      l.set_cursor_position(0,0)
```



It's important to test that the Displayotron works before you start the project. The best way is to write "Hello World" on the LCD screen.

Scrolling text is an old method of displaying information in a small amount of space, but you need to ensure the speed is just right.

```
    l.write(scrollBlurb[i:(i+16)])
    time.sleep(0.25)
else:
  l.set_cursor_position(0,0)
  l.write(scrollBlurb)
  print(scrollBlurb)
```

Our next function is called **scrollText**, and as its name suggests we use this to scroll text across the LCD screen. This function takes an argument which is the text to be scrolled. We start by checking the length of the text, and if it is greater than 16 characters then we enter an **If..Else** conditional statement, which for **if** creates a variable called **padding** that will contain 16 spaces. We then change the contents of the **scrollBlurb** variable so that it now contains the padding and text to scroll with an additional space at the end of the text.

Next we use a **for** loop that will iterate the same number of times as there are characters in the text to scroll. We write the text starting in the top-left of the screen, which is column 0 row 0. We then write the text starting with the character that is at position **i** in the text, where **i** is a an integer created by the **for** loop that increments by 1 each time the loop is run. We then add 16 to the **i** value to create an endpoint. The script then sleeps for 0.25 seconds before repeating. If the text is shorter than 16 characters then the **else** part of the conditional statement is true and the cursor position is set to the top-left of the screen, and the text is written to the LCD and printed to the shell for debug.

```
def player(radio):
  global p
  p = vlc.MediaPlayer(radio)
  p.play()
```

Our next function handles playback of the radio streams. We call the function **player** and it takes the argument **radio**, which will be the URL of the radio station that we wish to listen to. We use the global variable **p** and then store the radio station details ready for *VLC* to use as **p**. Next we instruct *VLC* to play the radio station stored in the variable **p**.

```
def stop():
  p.stop()
  b.rgb(255,0,0)
  l.write("S T O P")
  b.graph_off()
```

Our last function handles stopping any audio playback. It calls the **vlc.MediaPlayer(radio)** function to stop, then changes the LCD screen backlight to red before printing "S T O P" on the LCD and ensuring that any of the bar graph LED are turned off.

```
l.clear()
b.graph_off()
```

We now move to the main body of code, and we start by ensuring that the LCD is clear and that the bar

External sources of data

Pulling data from the internet is actually quite easy thanks to the many different APIs (Application Programming Interface) and many sources of information. So let's take a look at a few…

1) Pyjokes – http://pyjok.es
We all love a good joke and with this library you have access to many programming based jokes. You can install the library via **pip** with:

```
sudo pip install pyjokes
```

2) OpenWeatherMap – http://openweathermap.org
Do you need an umbrella or sunscreen? Well fear not: using Open Weather Map and **pyowm**, available from **https://pypi.python.org/pypi/pyowm/0.2.0** you can find out the weather for any location and it can be easily used in this project.

3) Fortune – https://pypi.python.org/pypi/fortune/1.0.1
Your own computerised fortune teller in Python, and very similar to the **fortune bash** command. When used it provides a random fortune or quote.

4) ihackernews – https://pypi.python.org/pypi/ihackernews/1.0.0 Even Hacker News has its own API.

5) narwal – https://pypi.python.org/pypi/narwal/0.3.2b
Now you can procrastinate on reddit using Python, or pull the latest news from your favourite subreddit.

There are many more Python libraries that can be used with your information centre and you can find out more from **https://pypi.python.org/pypi/pip**.

As for RSS feeds, there are many available covering multiple topics, but some of our favourites are:
BBC RSS Feeds – www.bbc.co.uk/news/10628494
NASA – www.nasa.gov/content/nasa-rss-feeds
Raspberry Pi Foundation Blog – www.raspberrypi.org/feed

The Displayotron's six touch-sensitive buttons will even work through acrylic, enabling projects to be enclosed.

graph LEDs are off.

```
@j.on(j.UP)
def handle_up(ch,evt):
  print("Playing BBC Radio 2")
  l.clear()
  b.rgb(255,0,255)
  l.write("BBC Radio 2")
  player(radio2)
```

Next we have blocks of code that handle the user pressing the capacitive touch keys located on the Displayotron HAT. To illustrate we will show how we can use the Up key, located on the left of the LCD, to play a radio station. We renamed **dothat.touch** as **j** when we imported the library, so we instruct the code to look for the up button to be pressed, and when that is true it prints the radio station to the Python shell for debug, clears the LCD screen, changes the backlight so that Red and Blue are at full brightness, writes "BBC Radio 2" to the LCD and then calls the player function with the argument **radio2**, which will instruct the player to load that station. The code for the down button is similar but replaces the radio station with BBC 6 Music and changes the backlight colour to the exact colour used for the station's branding.

```
@j.on(j.LEFT)
def handle_left(ch,evt):
  print("Left pressed!")
  l.clear()
  l.write("BBC News Feed")
  time.sleep(1)
  l.clear()
  b.rgb(0,0,128)
  feedme("http://feeds.bbci.co.uk/news/rss.xml")
```

The code for the left and right buttons is similar to up and down. We print to the Python shell that the button has been pressed, then we clear the LCD screen and write the name of the feed that we will be reading. This stays on the LCD for 1 second before the LCD screen is cleared. The backlight colour is changed to a light blue, and then the **feedme** function is called with the web address of the feed, in this case the BBC's top news stories. Pressing the right button will call another feed and change the colour of the backlight.

```
@j.on(j.BUTTON)
def handle_button(ch,evt):
  print("Button pressed!")
  l.clear()
  b.rgb(0,0,0)
  l.set_cursor_position(0,0)
  l.write("Linux Voice")
  for i in range(256):
    b.left_rgb(i,0,0)
    time.sleep(0.01)
    b.mid_rgb(i,0,0)
    time.sleep(0.01)
    b.right_rgb(i,0,0)
    time.sleep(0.01)
```

Our next function handles the button in between the left and right buttons. When this button is pressed we print that it is working to the Python shell. The LCD screen is cleared and the backlight is set to off. We then ensure that the LCD cursor position is in the top-left of the screen and then write "Linux Voice" to the screen. Next we use a **for** loop that will repeat 255 times, each time it loops each of the three LEDs that make up the backlight, have their value increased by one. A delay of 0.01 seconds is applied between each change in value and this creates a gradual glow that slowly fades to life.

```
@j.on(j.CANCEL)
def handle_cancel(ch,evt):
  print("Stopping Music")
  l.clear()
  b.rgb(0,0,0)
  stop()
```

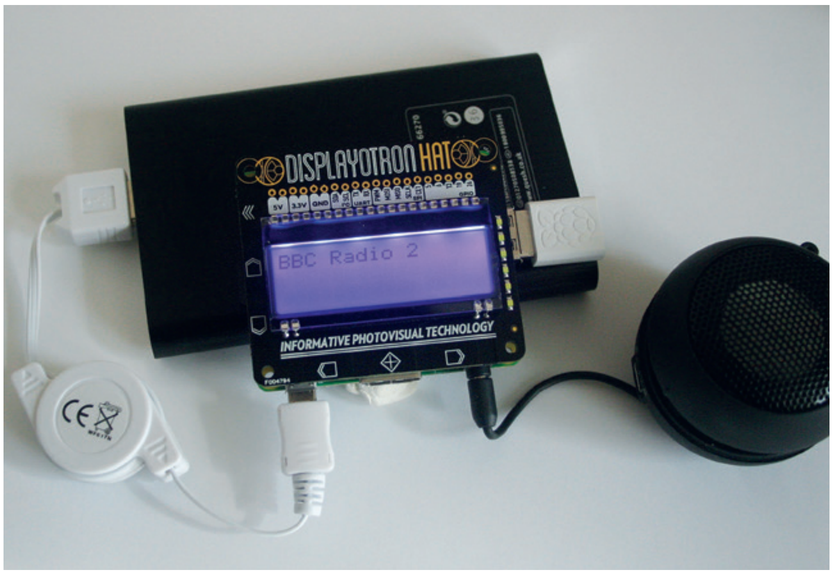
Finally we create a function that handles the button located at the top-left of the LCD screen. We shall use this to stop audio playback. Once the button is pressed, output for debugging is passed to the shell, the LCD screen is cleared and the backlight is turned off. We then launch the **stop** function and that handles turning off the audio and alerting the user.

```
signal.pause()
```

Our last line of code ensures that the project does not automatically shut down.

When you're ready, launch the code by clicking Run > Run Module. Press the up button to play a radio station, then press the cancel button on the top-left of the Displayotron to stop playback before pressing the down button. While the audio is playing you can press the left or right buttons to read the latest RSS feeds.

Congratulations! You have an information centre!



This project is well suited to the Raspberry Pi A+, as it's power efficient and cheap.

PRO TIP
All of the code for this project can be found via GitHub **http://bit.ly/LV21-Tutorial** or you can download the project as a ZIP file from **http://bit.ly/LV21-Tutorial-ZIP**.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

AUTOMATIC VULNERABILITY SCANNING WITH OPENVAS

Keeping track of security issues can be exhausting – so let your computer do it.

BEN EVERARD

WHY DO THIS?

- Find vulnerabilities on your servers
- Stop hackers breaking in
- Keep an ever-watchful robotic eye on your machines to make sure that they don't become vulnerable

If you run a server that's accessible from the internet, whether it's a home server or one running a multi-billion pound business, you need to make sure it's secure. If it's not, the it's just a matter of time before it falls prey to attackers. Keeping safe means always ensuring that your public-facing software is up to date and making sure it's configured correctly. In this tutorial, we're going to look at a piece of software that can automate this task – *Open Vulnerability Assessment Scanner* (*OpenVas*), which is a tool to run checks on servers to see if they have any weaknesses that attackers could exploit.

It can be a little tricky to set up *OpenVas* correctly, but to make it easier, the project provides a pre-configured virtual machine image that you can download. You can grab this from **www.openvas.org/vm.html**.

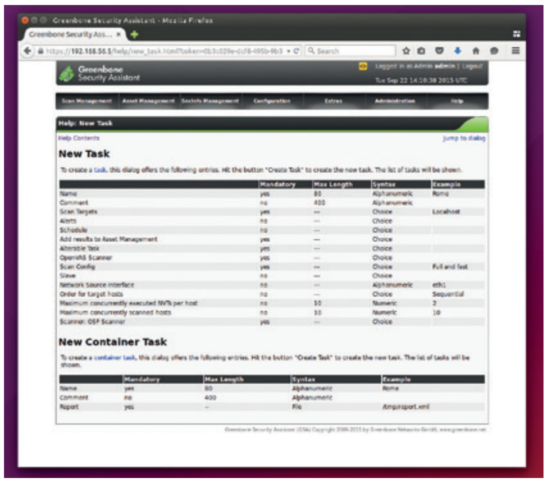
As well as *OpenVas*, we'll need a machine to scan for vulnerabilities. This could be a physical or a virtual machine, but it's easier to understand *OpenVas* if the machine has some security holes that the software can detect. We opted to use Metasploitable 2. This is a virtual machine that's deliberately insecure for practising attacks on. You can get this from **http://sourceforge.net/projects/metasploitable**.

Keeping safe means always ensuring that your public-facing software is up to date and configured correctly

To use these two virtual machines, you need to install *VirtualBox*, which you should find in your distro's repositories. Once you have this installed, you need to set up a host-only network. This will enable the two virtual machines to talk to each other, but won't expose either one to the network outside the machine. This is important because the exploitable machine is deliberately vulnerable and could easily be attacked if it were public.

First, create a host-only network by going to File > Preferences > Network > Host-only Networks and clicking Add. Then, highlight the newly created network (which will probably be called **bvoxnet0**), and click on the screwdriver icon. In the DHCP Server tab, make sure that Enable Server is checked and that all the boxes are populated. If they're not, enter 192.168.56.1, 255.255.255.0, 192.168.56.2 and 192.168.56.100 from top to bottom.

The *OpenVAS* file came as a *VirtualBox* appliance, so all you have to do is import the machine. Go to File > Import Appliance and select the OVA file that you've just downloaded, and it will create the virtual machine. The only thing you need to do is set the new machine to use the host-only network. Highlight the new machine, then press the settings button. In the


Greenbone Security Assistant - Mozilla Firefox
Greenbone Security Ass...
https://192.168.56.5/help/new_task.html?token=0b3c029e-dcf8-495b-9b3
Search
Greenbone Security Assistant
Logged in as Admin admin | Logout
Tue Sep 22 14:10:38 2015 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Administration
Help
Help: New Task
Help Contents
Jump to dialog

New Task

To create a task, this dialog offers the following entries. Hit the button "Create Task" to create the new task. The list of tasks will be shown.

	Mandatory	Max Length	Syntax	Example
Name	yes	80	Alphanumeric	Rome
Comment	no	400	Alphanumeric	
Scan Targets	yes	—	Choice	Localhost
Alerts	no	—	Choice	
Schedule	no	—	Choice	
Add results to Asset Management	yes	—	Choice	
Alterable Task	yes	—	Choice	
OpenVAS Scanner	yes	—	Choice	
Scan Config	yes	—	Choice	Full and fast
Slave	no	—	Choice	
Network Source Interface	no	—	Alphanumeric	eth1
Order for target hosts	no	—	Choice	Sequential
Maximum concurrently executed NVTs per host	no	10	Numeric	2
Maximum concurrently scanned hosts	no	10	Numeric	10
Scanner: OSP Scanner	yes	—	Choice	

New Container Task

To create a container task, this dialog offers the following entries. Hit the button "Create Task" to create the new task. The list of tasks will be shown.

	Mandatory	Max Length	Syntax	Example
Name	yes	80	Alphanumeric	Rome
Comment	no	400	Alphanumeric	
Report	yes	–	File	/tmp/report.xml

Greenbone Security Assistant (GSA) Copyright 2009-2015 by Greenbone Networks GmbH, www.greenbone.net

The *Greenbone* documentation covers almost everything you need to know, so if you ever get stuck, just drop into the Help menu and you should find out how to proceed.

Keeping OpenVAS updated

Network Vulnerability Tests (NVTs) are the scripts that *OpenVAS* uses to identify any security problems. They're constantly being updated, so if you want to check your machines against the very latest security issues, you need to make sure you have the latest NVTs. You can get these by running the following in a terminal on your *OpenVAS* box:

```
openvas-nvt-sync
```

This will grab the latest NVTs from the official *OpenVAS* feed. NSTs are written in NASL, a C like language that originated with the *Nessus* security scanner. You can create your own, although the process isn't particularly well documented. There's some information on how to get started writing NVTs on the *OpenVAS* website at **www.openvas.org/nvt-dev.html**.

Automatic vs manual pen testing

Vulnerability scanning with a tool like *OpenVAS* is a great way of improving the security of a server. It's quick, easy and cheap. However, it's not a simple solution to all security problems. The scans that *OpenVAS* runs try to identify known security vulnerabilities. There may be other issues that haven't been publicly reported that hackers are exploiting. Another limit of automatic vulnerability scanning is that software that's custom written won't be adequately checked for security holes. Because of all these reasons, a vulnerability scanner should never be seen as a guarantee that there aren't any issues on the server, and all normal security precautions should still be taken even if the scanner doesn't flag up any issues.

In contrast, a manual penetration test performed by an experienced tester should be far more thorough and ultimately lead to a more secure system than an automatic scanner. A good manual penetration test should identify areas of potential vulnerability that don't have published security holes (this could be particularly important if you have unusual configurations or custom code). It may also test out the effectiveness of mitigation tools such as intrusion detection systems that won't be tested using automatic scanners. *OpenVAS* can form a useful part of your security policy, but shouldn't be considered a substitute for having robust security.

new window, go to Network, and change Attached To to Host Only Adapter.

Metasploitable 2 comes as a virtual disk image rather than an appliance, so you have to manually create a virtual machine for it. Click New to open the new machine wizard, then enter a name and select Type: Linux and Version: Ubuntu (64 bit). The default 512MB of RAM is fine, as we won't be using this machine for anything taxing.

In the Hard Drive screen, select Use An Existing Hard Drive, then select the hard drive image that you've just downloaded and uncompressed. With this done, you can click on Create. The final thing needed to set up this virtual machine is changing the network adaptor to Host-Only in the same way as with the *OpenVAS* virtual machine.

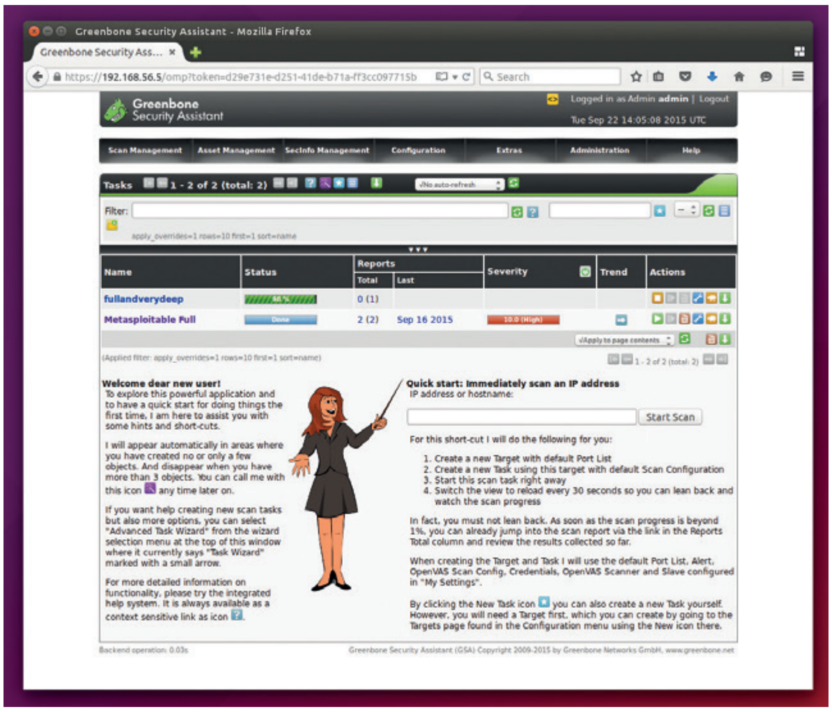
You can now start both the virtual machines by right-clicking on them and selecting Start. This will open new windows for the virtual machines, and you can interact with them through this window.

OpenVAS runs as a daemon, which enables you to schedule and queue jobs to run in whatever fashion you want. Happily, there's a web-based interface to *OpenVAS* called *Greenbone Security Assistant*, so we'll interact with this rather than *OpenVAS* itself.

Once the *OpenVAS* virtual machine has started, you will see something similar to:

```
The web interface is now available at: 192.168.56.2
```

The numbers at the end might be different when you run it. Whatever they are, this is the web address for the *Greenbone Security Assistant*, so open up a web browser and point it to http://192.168.56.2 (or whatever IP address is displayed in the virtual machine).


Greenbone Security Assistant - Mozilla Firefox
Greenbone Security Ass...
https://192.168.56.5/omp?token=d29e731e-d251-41de-b71a-ff3cc097715b
Greenbone Security Assistant
Logged in as Admin admin | Logout
Tue Sep 22 14:05:08 2015 UTC
Scan Management | Asset Management | SecInfo Management | Configuration | Extras | Administration | Help
Tasks 1 - 2 of 2 (total: 2) √No auto-refresh
Filter:
apply_overrides=1 rows=10 first=1 sort=name

Name	Status	Reports Total	Reports Last	Severity	Trend	Actions
fullandverydeep	98 %	0 (1)				
Metasploitable Full	Done	2 (2)	Sep 16 2015	10.0 (High)		

√Apply to page contents
(Applied filter: apply_overrides=1 rows=10 first=1 sort=name)
1 - 2 of 2 (total: 2)

Welcome dear new user!
To explore this powerful application and to have a quick start for doing things the first time, I am here to assist you with some hints and short-cuts.

I will appear automatically in areas where you have created no or only a few objects. And disappear when you have more than 3 objects. You can call me with this icon any time later on.

If you want help creating new scan tasks but also more options, you can select "Advanced Task Wizard" from the wizard selection menu at the top of this window where it currently says "Task Wizard" marked with a small arrow.

For more detailed information on functionality, please try the integrated help system. It is always available as a context sensitive link as icon.

Quick start: Immediately scan an IP address
IP address or hostname:
Start Scan

For this short-cut I will do the following for you:

1. Create a new Target with default Port List
2. Create a new Task using this target with default Scan Configuration
3. Start this scan task right away
4. Switch the view to reload every 30 seconds so you can lean back and watch the scan progress

In fact, you must not lean back. As soon as the scan progress is beyond 1%, you can already jump into the scan report via the link in the Reports Total column and review the results collected so far.

When creating the Target and Task I will use the default Port List, Alert, OpenVAS Scan Config, Credentials, OpenVAS Scanner and Slave configured in "My Settings".

By clicking the New Task icon you can also create a new Task yourself. However, you will need a Target first, which you can create by going to the Targets page found in the Configuration menu using the New icon there.

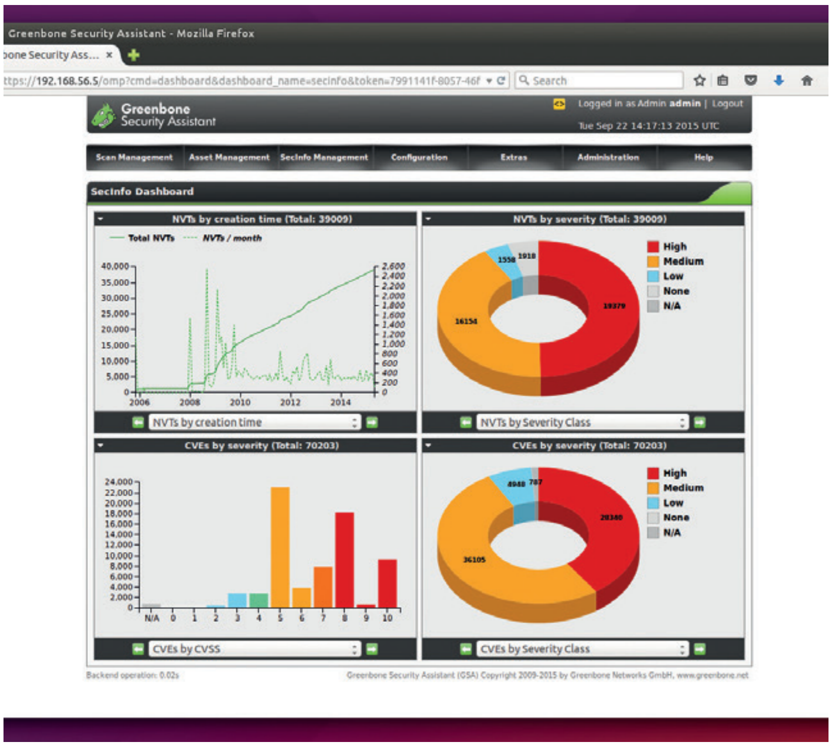
Backend operation: 0.03s
Greenbone Security Assistant (GSA) Copyright 2009-2015 by Greenbone Networks GmbH, www.greenbone.net

There is a wizard that's supposed to ease the task of creating tasks, but we find it easier to build them manually, as the wizard can be a little confusing.

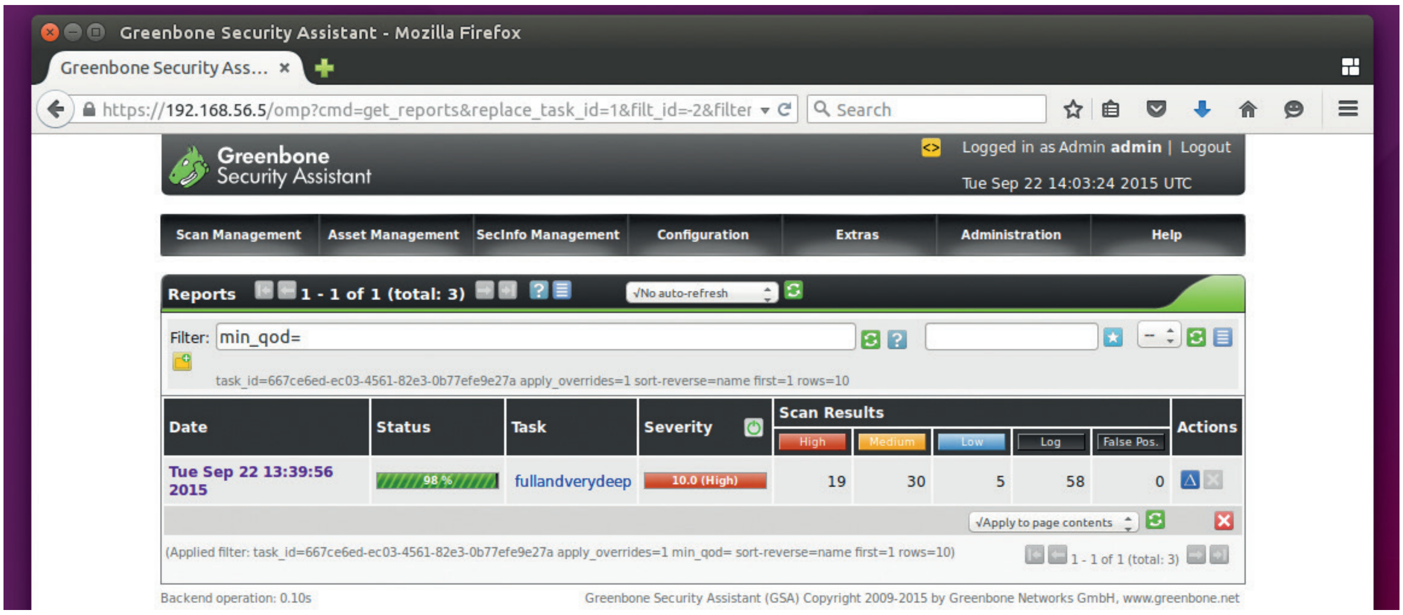
You'll then need to log in. The username and password are both admin.

Greenbone

The user interface of the *Greenbone Security Assistant* leaves a little to be desired, but provided you do everything in the right order, it works well. The interface revolves around targets, tasks and reports. Targets are machines (usually identified by IP address) that you wish to scan. Tasks are a particular setup of target and type of scan that can be run one or more times. Reports are the results of


Greenbone Security Assistant - Mozilla Firefox
bone Security Ass...
ttps://192.168.56.5/omp?cmd=dashboard&dashboard_name=secinfo&token=7991141f-8057-46f
Search
Greenbone Security Assistant
Logged in as Admin admin | Logout
Tue Sep 22 14:17:13 2015 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Administration
Help
SecInfo Dashboard
NVTs by creation time (Total: 39009)
Total NVTs
NVTs / month
NVTs by severity (Total: 39009)
High
Medium
Low
None
N/A
1558
1918
19379
16154
NVTs by creation time
NVTs by Severity Class
CVEs by severity (Total: 70203)
CVEs by severity (Total: 70203)
4948
787
28340
36105
CVEs by CVSS
CVEs by Severity Class
Backend operation: 0.02s
Greenbone Security Assistant (GSA) Copyright 2009-2015 by Greenbone Networks GmbH, www.greenbone.net

The SecInfo Dashboard will give you details of all the vulnerabilities and tests that *OpenVAS* knows about.


Greenbone Security Assistant - Mozilla Firefox
Greenbone Security Ass...
https://192.168.56.5/omp?cmd=get_reports&replace_task_id=1&filt_id=-2&filter
Search
Greenbone Security Assistant
Logged in as Admin admin | Logout
Tue Sep 22 14:03:24 2015 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Administration
Help
Reports 1 - 1 of 1 (total: 3)
√No auto-refresh
Filter: min_qod=

task_id=667ce6ed-ec03-4561-82e3-0b77efe9e27a apply_overrides=1 sort-reverse=name first=1 rows=10

Date	Status	Task	Severity	Scan Results					Actions
				High	Medium	Low	Log	False Pos.	
Tue Sep 22 13:39:56 2015	98 %	fullandverydeep	10.0 (High)	19	30	5	58	0	

(Applied filter: task_id=667ce6ed-ec03-4561-82e3-0b77efe9e27a apply_overrides=1 min_qod= sort-reverse=name first=1 rows=10)

1 - 1 of 1 (total: 3)

Backend operation: 0.10s

Greenbone Security Assistant (GSA) Copyright 2009-2015 by Greenbone Networks GmbH, www.greenbone.net

You can view the content of the report before the scan is finished, which is particularly useful for long-running, complex scan profiles like Full and Very Deep.

tasks running. The process of scanning a machine is creating the target, creating the task, scheduling the task to run and then investigating the report.

The first task, then, is to create a target. Targets are basically names given to IP addresses, so we need to find out the IP address of the Metasploitable virtual machine. Switch to the Metasploitable window and log in with msfadmin/msfadmin and enter:

```
ip addr
```

In the output, you should see a line that looks something like this:

```
inet 192.168.56.3/24
```

The numbers may be different, but this (192.168.56.3) is the IP address of the Metasploitable machine. In *Greenbone*, you can now go to Configuration/Targets. The Add New icon is a white star on a blue background. It's small and easy to miss, but once you've found it, it's in the same place on all the *Greenbone* screens. Click on this to create a new Target.

Give the target a name (we went with the rather unimaginative 'Metasploitable'), make sure Manual is checked and enter the IP address in the host box. With this done, just click Create Target.

The next step is to create a task. These are scan configurations that can be executed. In normal use (which we'll look at a bit later), tasks can be run at regular intervals to make sure that a given machine stays secure.

Go to Scan Management > Tasks and click on the new icon to set up a task. The key items on this form are the Name (this can be anything that you use to identify the scan), the Scan Target, which should be the target that you've just set up, and the Scan Config. This final item dictates the type of scan being run. It comes down to which checks are run against the host. The more checks that are run, the more thorough the scan will be, but the longer it will take and the more noise it will generate on the network (see boxout on Automatic and Manual Testing). We'll go for a Full and Fast test.

Testing for exploits

Click on Create Task (in the New Task box, not in the New Container Task box), and you'll go through to the Task screen. Here you should see all the details of the task that you've just created, but it won't yet have run so there aren't any results. Click on the green play icon at the top to start the task running straight away.

The page will hold the progress of the scan, but may not automatically update. Click on the refresh icon in the page (not the main browser refresh, but the icon on the web page), or change No Auto Refresh. The status entry will update to show you how far the scan has progressed. It may take a little while to complete, but once it has, you'll be able to view the report.

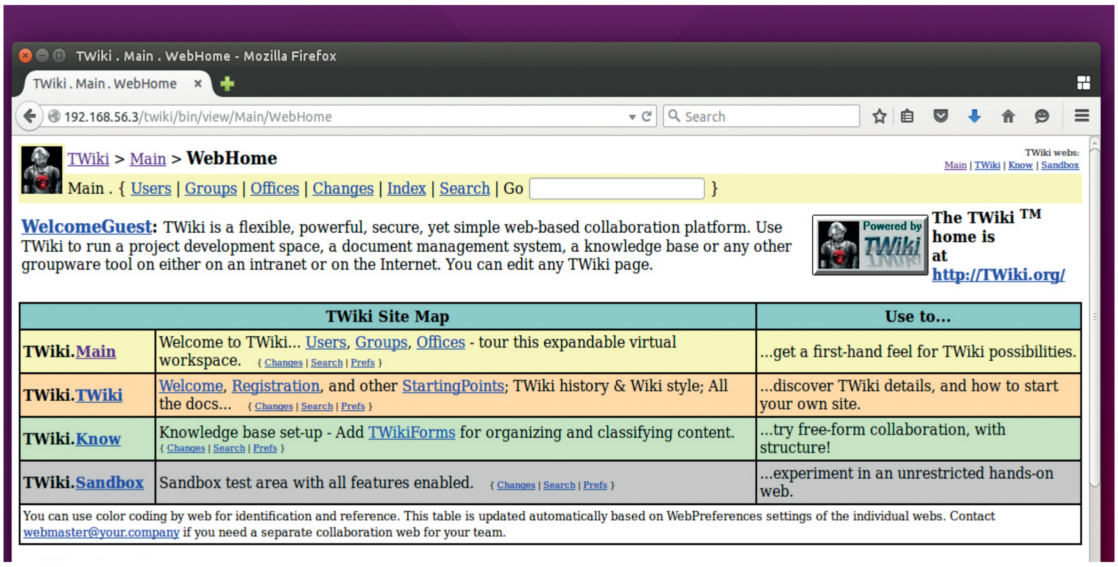
In Scan Management > Reports, you'll see a list of every task that has run. At the moment, it will just

You can run periodic scans against your machines and receive the reports detailing how they have performed

Alternative vulnerability scanners

OpenVAS is a fork of the *Nessus* security scanner. *Nessus* was originally released under the GPL, but its developers decided to close the source code (since they owned the copyright, they could legally stop releasing it under the GPL, but couldn't revoke the open source access to earlier code). *Nessus* remains a very popular scanner and is free (as in zero cost, not as in free software) for personal use.

There many other commercial vulnerability scanners available, including *Nexpose* from Rapid 7, and there are some specialised open source scanners that are good at identifying problems with particular pieces of software such as *WPScan* (**http://wpscan.org**), which checks *WordPress* sites for vulnerabilities, and OWASP's *ZAP*, which can check websites for potential security issues.



TWiki . Main . WebHome - Mozilla Firefox

192.168.56.3/twiki/bin/view/Main/WebHome

TWiki > Main > WebHome

TWiki webs: Main | TWiki | Know | Sandbox

Main . { Users | Groups | Offices | Changes | Index | Search | Go }

WelcomeGuest: TWiki is a flexible, powerful, secure, yet simple web-based collaboration platform. Use TWiki to run a project development space, a document management system, a knowledge base or any other groupware tool on either on an intranet or on the Internet. You can edit any TWiki page.

The TWiki ™ home is at http://TWiki.org/

TWiki Site Map		Use to...
TWiki.Main	Welcome to TWiki... Users, Groups, Offices - tour this expandable virtual workspace. { Changes | Search | Prefs }	...get a first-hand feel for TWiki possibilities.
TWiki.TWiki	Welcome, Registration, and other StartingPoints; TWiki history & Wiki style; All the docs... { Changes | Search | Prefs }	...discover TWiki details, and how to start your own site.
TWiki.Know	Knowledge base set-up - Add TWikiForms for organizing and classifying content. { Changes | Search | Prefs }	...try free-form collaboration, with structure!
TWiki.Sandbox	Sandbox test area with all features enabled. { Changes | Search | Prefs }	...experiment in an unrestricted hands-on web.

You can use color coding by web for identification and reference. This table is updated automatically based on WebPreferences settings of the individual webs. Contact webmaster@your.company if you need a separate collaboration web for your team.

Metasploitable includes a variety of vulnerable web applications including TikiWiki. Take a look at the OpenVAS report for details of how to attack them.

show a single entry. In this screen, you can see a summary of the results breaking down any issues it finds into high, medium and low severity. There are also items listed as Log, which means that there isn't a vulnerability, but information that you may wish to look at. In this report, there should be lots of each level of issue. Click on the date to get the detailed contents of the report.

If this were a scan on a machine that was network-accessible, all the entries listed should be fixed. If you click on an entry in the Vulnerability column, then you'll see more details about the particular issue. This Vulnerability Details screen holds a few key pieces of information. As well as a general description of the problem, you should see details of how to fix the issue (typically, this is upgrading the problem software, but it could be filtering ports in the firewall or changing configurations). There should also be a CVE reference. This is an identification number in the Common Vulnerabilities and Exposures database, which is accessible at **https://cve.mitre.org**. This database enables security researchers and software developers to track any problems. They're particularly useful for identifying which release of a program fixes a specific issues, as the CVE should be in the release notes. There will often be a link to the software's website that should have details about how the vulnerability can be fixed.

Schedules and alerts

One of the great features of *Greenbone* is the ability to schedule scans. Using this, you can run periodic scans against your machines and receive the reports detailing how the machines have performed. You can even set up the system so that it runs the scans silently and only alerts you when vulnerabilities are discovered. These features are set up using Schedules and Alerts.

In OpenVAS, schedules and alerts are both entities that are created and then added to the task. This is a little counter intuitive, since it's more natural to think of a schedule as something to which you add a task, rather than the other way around. As long as you do everything in the right order, though, everything is relatively easy.

Both schedules and alerts have entries in the Configuration menu that take you to the appropriate forms, which are easy to fill in. Once you've entered the details you need, you can go back to the task (Scan Management > Tasks and then click on the appropriate task), and use the spanner icon to edit the task. In the task, you can apply whichever schedule and alerts you wish. Using this, you can make sure that servers don't regress into unsafe states or miss out on critical security updates. And that's about it – automatic vulnerability scanning, made easy!

Ben Everard is the best-selling co-author of the best-selling book, *Learning Python with Raspberry Pi*.

Exploiting Metasploitable

If you run a scan of Metasploitable using the profile Full And Very Deep Ultimate, it will check very thoroughly for vulnerabilities. In fact, it should find 19 high-priority issues. Let's take a look at just how serious some of these are. One of the severity 10 issues in the report is NFS Export. If you click on that, you'll see that the problem is that the root directory (/) is being shared using the Network File System (NFS). At you might imaging, this is quite a serious issue that can be exploited very easily.

To see how a hacker can break in, you'll first need to make sure that you have the relevent **nfs** tools installed on your machine. On an Ubuntu-based machine, this is done with the command:

```
sudo apt-get install rpcbind nfs-common
```

Once you've got these, you can create a new directory and mount the exposed network share with:

```
mkdir /tmp/nfs-test
sudo mount.nfs 192.168.56.3:/ /tmp/nfs-test/
```

After this, you should find that you have complete access to Metasploitable's filesystem. Not all of the vulnerabilities are as serious as this, but there are many serious problems that can allow hackers to gain access to the system.

BUILD A 3D HEAD TRACKER WITH OPENTRACK

Map your head movements to mouse control with a handful of open source.

GRAHAM MORRISON

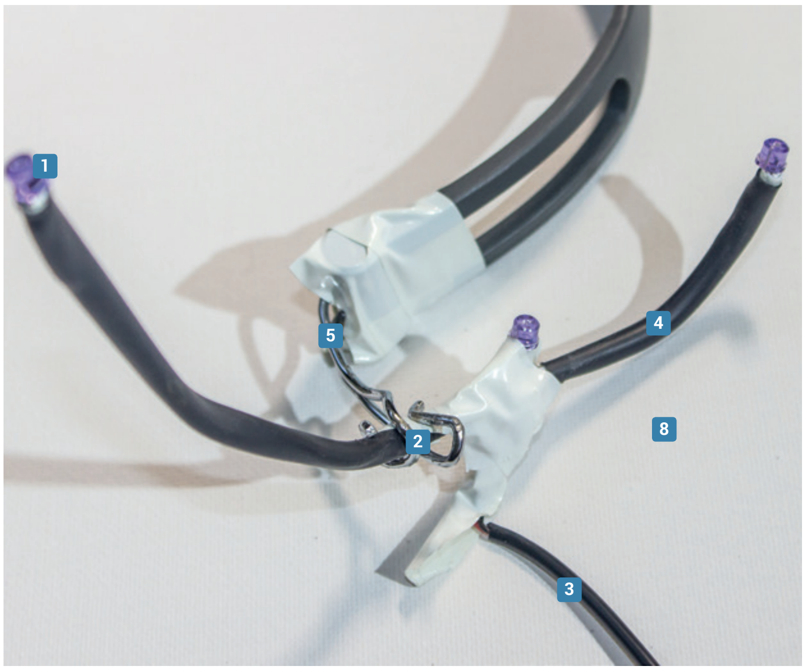
WHY DO THIS?

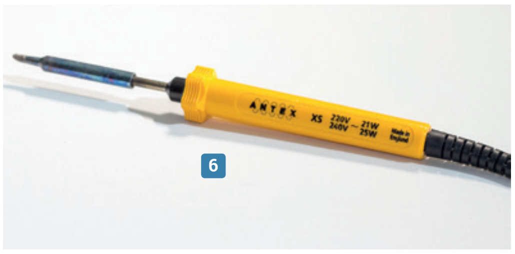
- It's cheap and very cool
- Generates X, Y, Z, yaw, pitch and roll data
- Augment virtual reality…
- …or play games with your head

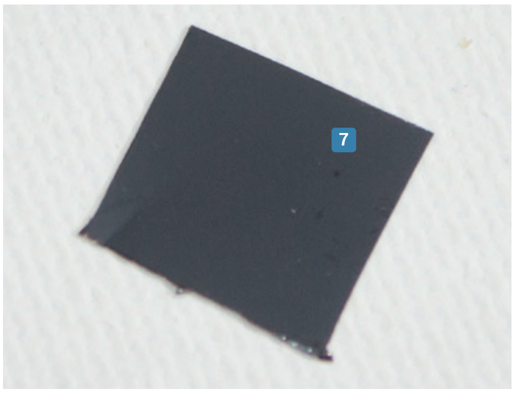
After experimenting with the kind of gyro sensors that festoon most smartphones, the Oculus Rift virtual reality headset, due to be released in Q1 2016, opted instead for a series of infrared LEDs surrounding the headset itself, coupled with external cameras that translate the movement of those LEDs into 3D space. This is because it's often quicker and more accurate, especially when cameras operate at 60 or 100 frames per second. The Oculus Rift is cutting edge, but the principles behind this tracking have been around for a while, especially among gamers where head tracking provides an additional level of immersion in first-person games. Turn your head left, for example, and your view turns to the left.

But there's also a great open source project (called *Opentrack*) that uses a DIY headset and translates head movement into up/down (y), left/right (x), in/out (x), roll, pitch and yaw parameters, which you can then use for anything from playing games to controlling your desktop. And it's exactly this kind of low-cost configuration we're going to build now.

What you'll need







1 3 x infrared LEDs
We bought OSRAM SFH485P LEDs from Farnell. Because they radiate light at infrared range, they're not visible to the human eye.

2 1 x 6.8Ω resistor
The circuit for powering the LEDs is incredibly simple. You need a single resistor and some wire. We're taking 5v power from a USB power adaptor.

3 USB lead
We also butchered a USB lead so we can connect the headset to a USB port on a computer or a phone charger, but batteries are another option.

4 6.4mm heat shrink
This stuff is amazing. Put your wires and even resistors into the tubing and heat it up with a hair dryer and it shrinks around the entire package.

5 Metal coat hanger/wire
There are may ways to construct a frame. Some projects have used pen cases. But we've had good results with a wire hanger.

6 Soldering iron/solder
We're only going to make a few soldering points – connecting the LEDs to wire and the resistor, making this project ideal for a beginner.

7 Visible light filter
Yes, we've used a cutout from an old 3.5-inch floppy disc. Any old floppy will do but you get extra bonus points for using Amiga Format's Cannon Soccer coverdisc. You could also used an exposed piece of old school negative film.

1 Which webcam

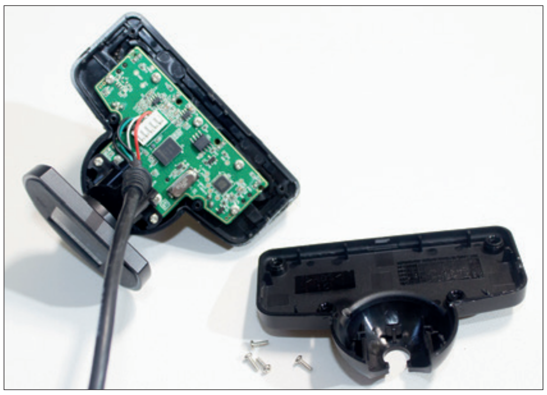
The Playstation 3 webcam we've chosen is commonly used for head tracking and works perfectly with Linux. It's also easy to take apart, which we need to do so that we can remove the infrared blocking filter and add our infrared bandpass filter (blocking visible light), which can be slotted in front of the lens, blocking everything but infrared light from getting to the sensor.

It's also cheap, with new models selling for around £4 delivered on eBay. But the best thing about this camera is that it's capable of 100 frames per second, and you can change the field of view by twisting the lens surround. Unfortunately, the process of taking the camera apart and placing a filter across the lens renders it useless as a normal webcam, unless you're happy capturing out of focus images of infrared light.

2 Disassemble webcam

We're now going to do some physical hacking. Taking apart the webcam is relatively easy, although you do need to use a little brute force at one point. Start with the small rubber covers hiding the four screws on the rear, followed by the screws themselves.

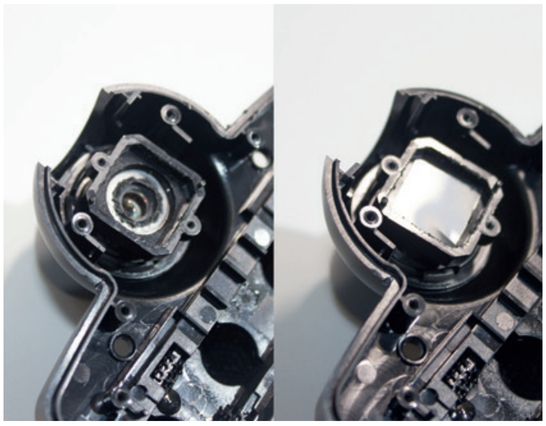
The front and rear now need to be pried open with a flat-headed screwdriver. Be careful as the main PCB is held in between these two parts and you'll need some patience as you work around the gap, gently making it larger until it comes apart. You'll now see the PCB, and you need to remove the screws that connect the PCB to the internal plastic. The PCB should then come out without difficulty, revealing the lens component, which can be taken off after removing the two screws holding it on.



3 Removing and adding the filter

Be careful not to smudge the lens or the sensor, or to allow dust to settle on the sensor. There are now two steps to perform before the lens will allow only the infrared light from our LEDs. The first is to remove the infrared blocking filter, which is the thin sliver of glass closest to the sensor in the lens. This is best removed with a sharp knife, going around the edge of the filter before levering the glass from inside.

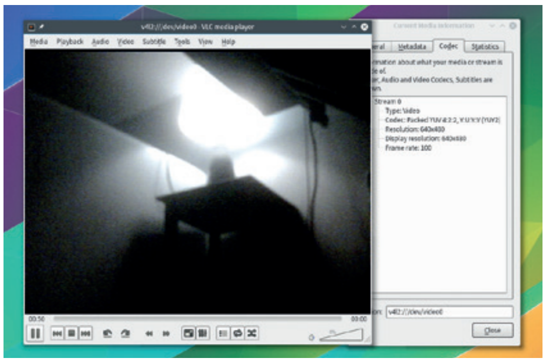
We now want to replace this with our bandpass filter, allowing only the light from the LEDs. We took apart an old Amiga 3.5-inch floppy disc and cut a small section that would fit exactly into the gap left by the blocking filter. You then need to retrace your steps, putting the entire camera back together.



4 Testing the webcam

The Playstation 3 Eye camera works in Linux without any drivers. Just connect the camera's cable to a spare USB port. We used *VLC*, selecting Open Capture Device as a media source and making sure the Video Device Name is pointed to the webcam. If the infrared filter is working, the image will have washed out black and white appearance. You may notice that the camera only finds a focus somewhere in the middle of switching between the two field-of-view modes because removing the infrared blocking filter has changed the focal length of the lens. There are two solutions; the first is to do nothing, as even without focus the LEDs will be detected: the second is to carefully twist the lens until it sticks to the focus point.

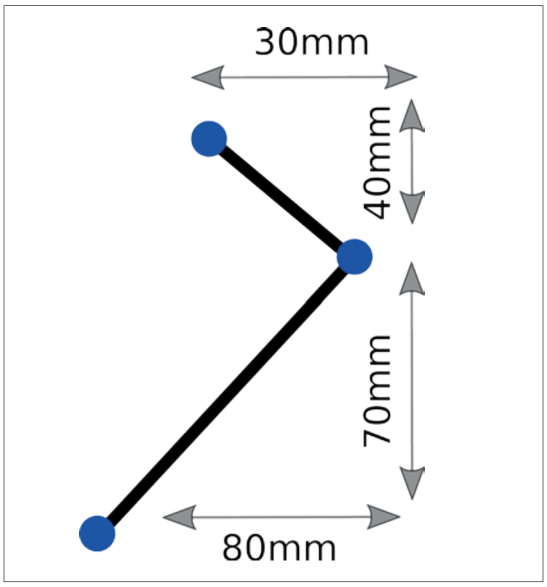
PRO TIP
Use the V4L2 Test Bench utility to change webcam parameters, such as contrast, brightness, exposure and frame rate, to improve the tracking algorithm.


v4l2:///dev/video0 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Current Media Information
Metadata
Codec
Statistics
Close

Stream 0
Type: Video
Codec: Packed YUV 4:2:2, Y:U:Y:V (YUY2)
Resolution: 640x480
Display resolution: 640x480
Frame rate: 100

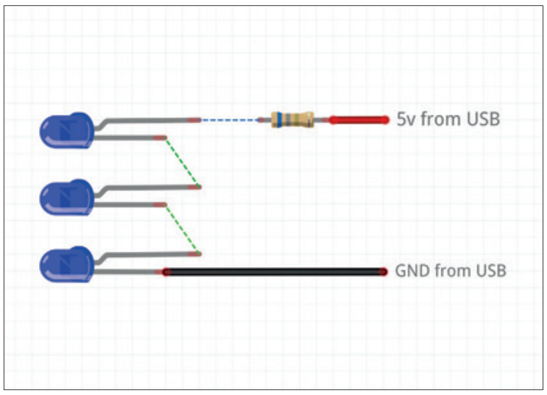
5 Tracking design

With the webcam working, it's now time to construct the frame that will hold our LED circuit; this is because the length of the wires you use will depend on the shape of the frame. Its optimal shape has already been calculated, and you should be able to see the dimensions below. It's an L shape, and two LEDs will sit on the ends while a third is at the corner of the shape. We'd recommend laying out the components and the wires so that they all sit close together.



6 Connect LEDs via wire and resistor

This couldn't be a simpler circuit. Cut open the spare USB cable and strip the ends of the red (+5v) and black (GND) wires. This will give our unit power. Connect the red +5v wire to one leg of the resistor and the other leg of the resistor to the shorter leg of one of the LEDs. LEDs have polarity, which means they need to be connected the correct way around. The short leg is usually positive while the short leg (along with a flat side on the LED) is usually negative. Connect the short leg of this first LED to the long leg of the next and do the same for the final LED, connecting its short negative to the GND (black) wire on the USB cable to complete the circuit. You can quickly check it works by connecting to a USB port and viewing using a smartphone camera.

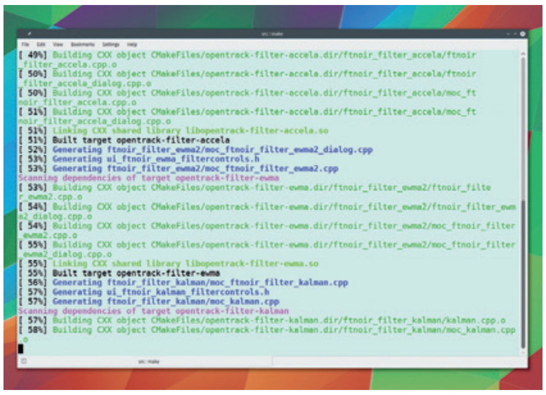


7 Shrink

With the circuit built and connected, and the basic design of the headset complete, it's time to put it all together. Our head tracking design is for a clip that's intended to connect vertically to the side of a cap, or best of all, a pair of headphones. It's easy to use a large clip to temporarily connect the headset, but we decided to dismantle an old pair of headphones and permanently attach the head strap to the LED head tracking device. It's worth putting the camera above your screen and positioning yourself with the headset to make sure all three LEDs are visible as you turn your head left/right and up/down. The orientation of the headset is such that the corner of the L shape sits near your ear, with the long edge going down to your chin and the short edge up to your eyebrow.

8 Install Opentrack

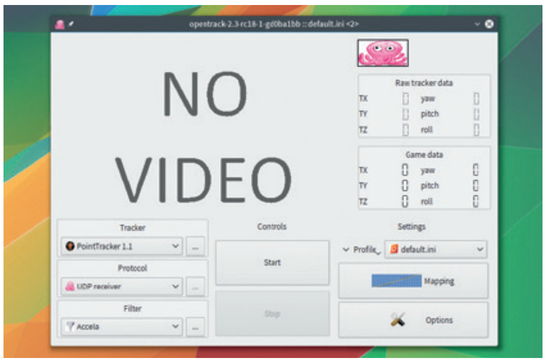
The software that we're going to use to take the realtime video of our headset and convert the movement of the LEDs into raw data is called *Opentrack*. You may be able to find a package for your distribution, but we needed to compile the latest release for Arch, which has a few dependencies, including **Qt5** and **Qt5-serial port**, **cmake** and **opencv**. To build the project, download the source code from **https://github.com/opentrack**. Unzip the file and **cd** into the directory, then type **mkdir build** to create a directory called **build** and **cd** into this. Now type **cmake ..** to generate a Makefile from the source code; when this has finished, type **make** to start the build process. When complete, you can run the binary from the build folder by typing **./opentrack.bin**.



```
[ 49%] Building CXX object CMakeFiles/opentrack-filter-accela.dir/ftnoir_filter_accela/ftnoir
_filter_accela.cpp.o
[ 50%] Building CXX object CMakeFiles/opentrack-filter-accela.dir/ftnoir_filter_accela/ftnoir
_filter_accela_dialog.cpp.o
[ 50%] Building CXX object CMakeFiles/opentrack-filter-accela.dir/ftnoir_filter_accela/moc_ft
noir_filter_accela.cpp.o
[ 51%] Building CXX object CMakeFiles/opentrack-filter-accela.dir/ftnoir_filter_accela/moc_ft
noir_filter_accela_dialog.cpp.o
[ 51%] Linking CXX shared library libopentrack-filter-accela.so
[ 51%] Built target opentrack-filter-accela
[ 52%] Generating ftnoir_filter_ewma2/moc_ftnoir_filter_ewma2_dialog.cpp
[ 53%] Generating ui_ftnoir_ewma_filtercontrols.h
[ 53%] Generating ftnoir_filter_ewma2/moc_ftnoir_filter_ewma2.cpp
Scanning dependencies of target opentrack-filter-ewma
[ 53%] Building CXX object CMakeFiles/opentrack-filter-ewma.dir/ftnoir_filter_ewma2/ftnoir_filte
r_ewma2.cpp.o
[ 54%] Building CXX object CMakeFiles/opentrack-filter-ewma.dir/ftnoir_filter_ewma2/ftnoir_filter_ewm
a2_dialog.cpp.o
[ 54%] Building CXX object CMakeFiles/opentrack-filter-ewma.dir/ftnoir_filter_ewma2/moc_ftnoir_filter
_ewma2.cpp.o
[ 55%] Building CXX object CMakeFiles/opentrack-filter-ewma.dir/ftnoir_filter_ewma2/moc_ftnoir_filter
_ewma2_dialog.cpp.o
[ 55%] Linking CXX shared library libopentrack-filter-ewma.so
[ 55%] Built target opentrack-filter-ewma
[ 56%] Generating ftnoir_filter_kalman/moc_ftnoir_filter_kalman.cpp
[ 57%] Generating ui_ftnoir_kalman_filtercontrols.h
[ 57%] Generating ftnoir_filter_kalman/moc_kalman.cpp
Scanning dependencies of target opentrack-filter-kalman
[ 57%] Building CXX object CMakeFiles/opentrack-filter-kalman.dir/ftnoir_filter_kalman/kalman.cpp.o
[ 58%] Building CXX object CMakeFiles/opentrack-filter-kalman.dir/ftnoir_filter_kalman/moc_kalman.cpp
.o
```

9 Configuring Opentrack

Opentrack uses three different sets of protocols, each of which is selected and configured using the drop-down menus on the left. Tracker is the plugin that generates the raw data, Filter is the plugin that smooths out the rough edges from the data, and Protocol is the plugin that delivers this data to its final destination. We're going to use the *PointTracker 1.1* tracker to generate the data, as this is the one that derives positional data from the video of our LEDs, but there's also a plugin for an Arduino-based sensor if you wanted to take the project further. We've found the best filter plugin is *Accela*, which is usually selected by default, and the easiest protocol to get working is **libevdev joystick receiver**, which turns all of your movements into a virtual joystick that can then be reconfigured to control anything.

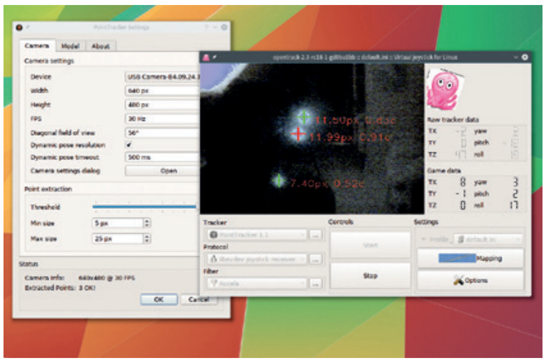


Tracker: PointTracker 1.1
Protocol: UDP receiver
Filter: Accela
Profile: default.ini

10 PointTracker 1.1

The only plugin that needs configuration is *PointTracker*. To do this, click on the **...** button to open the options view. First make sure your USB camera is selected in Device and click on Start in the main application window. This will start the tracking and the video input, enabling you to fine-tune the plugin while seeing the output. Natural light is a big source of infrared, so you may need to close curtains or doors until your LEDs are the only sources visible.

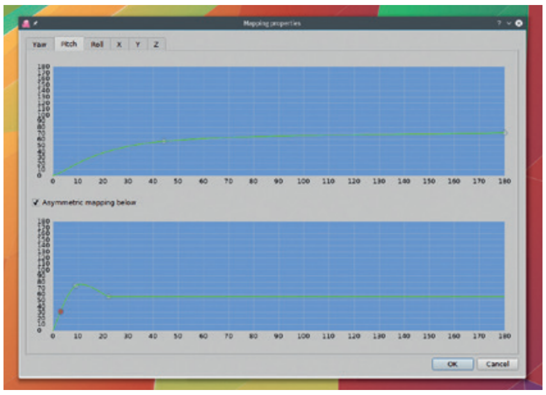
The main parameters to adjust are Threshold, which governs the strength of infrared declared a point, and min/max sizes. We set the threshold to almost its maximum, shielding out background light, and a min/max size of 5x/25px. When the tracking is working, you'll see the game data and the octopus image move in sequence with your own movements.



Device: USB Camera-B4.09.24.1
Width: 640 px
Height: 480 px
FPS: 30 Hz
Diagonal field of view: 56°
Dynamic pose resolution: ✓
Dynamic pose timeout: 500 ms
Camera settings dialog: Open
Min size: 5 px
Max size: 25 px
Camera info: 640x480 @ 30 FPS
Extracted Points: 3 OK!

11 Mapping

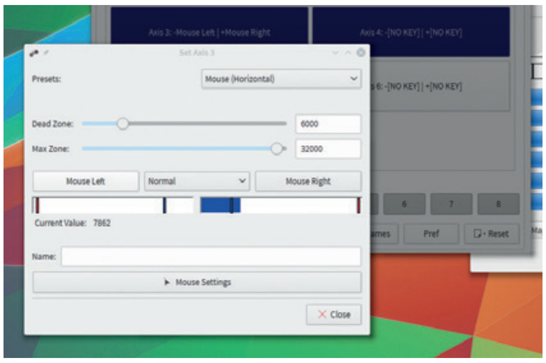
Before all this data becomes useful, you need to fine-tune the scale of each axis. This is done by clicking on the 'Mapping' button of the main application. With the tracker running, you'll see a cursor move along a line. Twist your head to the left, for example, and the roll angle will change, look down at the keyboard, or left and right, and pitch and yaw will change. The straight line can be edited to amplify the angle by dragging down the top-right point, and new Bézier points can be used to fine-tune the curve. By default, the curves are identical when you cross over the central point, but you make them asymmetric by enabling the mapping curve below. There's no correct way of doing this, but play with the curves until you can make best use of your movement.



12 Taking it further

Another essential step is to map 'Center' to a key from the Options panel in the main *Opentrack* window. This enables you to set your straight ahead central point at any time, effectively resetting the tracking data. The simplest way to play with the output is to use the joystick driver in your favourite game – manually mapping the input to the controls used to change the first-person view. It's also possible to map these inputs into desktop controls, moving the mouse around and even selecting things with a gesture. We used a tool called *jtest-gtk* to calibrate the inputs, and *antimicro* to map the inputs to mouse and keyboard controls. Finally, you can map the control to an external virtual reality headset using the 'UDP receiver' protocol.

PRO TIP
The Model view in the *PointTracker* plugin can be used to fine-tune the size of your LED clip and its position relative to the camera.


Axis 3: -Mouse Left | +Mouse Right
Axis 4: -[NO KEY] | +[NO KEY]
Set Axis 3
Mouse Left
Mouse Right
Mouse Settings
Close
Pref
Reset

Presets: Mouse (Horizontal)
Dead Zone: 6000
Max Zone: 32000
Normal
Current Value: 7862
Name:

CALCULATE YOUR BUS FACTOR WITH GIT AND R

Discover what would happen if open source lost some of its core developers.

ROBIN GOWER

WHY DO THIS?

- Find out if the kernel could cope without Torvalds...
- Compare your work with successful open source projects
- Read between the lines – learn about your codebase with data analysis

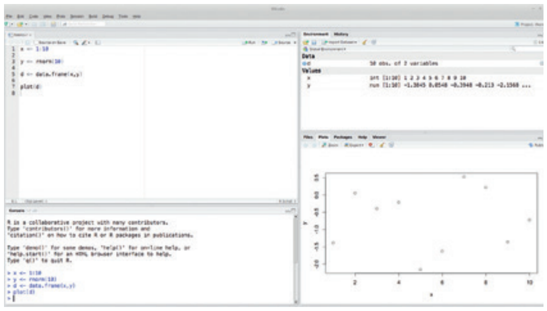
All open source projects rely on their developers, but some rely on a handful of people so much that the project would be in serious jeopardy if one or two people stopped coding. The degree to which a project is reliant on just a few developers can be measured by something called the bus factor, which is the number of developers that would need to be hit by a bus before a project is in serious trouble. A low bus factor means that knowledge is concentrated in a few vital people, while A high bus factor means that a broad range of individuals know enough to carry on with the project even if others leave.

In this tutorial we'll show you how to calculate the bus factor of your project, or of the open source projects you love (and perhaps depend upon). We'll get our data from *Git* commit logs and analyse it using GNU R, the statistical language and computing environment. To set-up an R environment you'll first need to download R, either from your distro's package manager or from **r-project.org**. R comes with its own REPL (read, evaluate, print, loop) interpreter, which you can start by running the command **R** at your prompt. However, instead of using this, we recommend you use an IDE. In recent years *RStudio* has established itself as the *de facto* standard (**rstudio.com**). In addition to the console you get an editor with syntax highlighting, plotting windows, a help browser and other tools. Once you're up and running you can install the libraries you'll need with R's package manager.

> The degree to which a project is reliant on a few developers can be measured by something called the bus factor

The RStudio IDE

Assuming you've also already installed the *Git* client from your distro's package manager, we're ready to go. We'll start with a brief tour of the *RStudio* interface. In the default layout you'll have four panes. In the top-left we have the source pane for editing code and sending it to the R interpreter displayed in the console beneath: press Ctrl+Enter to run the current line, or Ctrl+Shift+Enter to run the whole file (see the Code menu for related short-cuts).



```
x <- 1:10
y <- rnorm(10)
d <- data.frame(x,y)
plot(d)
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> x <- 1:10
> y <- rnorm(10)
> d <- data.frame(x,y)
> plot(d)
>
```

```
Data
d        10 obs. of 2 variables
Values
x        int [1:10] 1 2 3 4 5 6 7 8 9 10
y        num [1:10] -1.3845 0.0540 -0.3948 -0.213 -2.1568 ...
```

R Studio (AGPLv3) is a popular user interface built with the *Qt* and *GWT* graphical toolkits

Let's try this now. Create a new script (File > New File > R Script) and type **1:5**, with your cursor still on this line, hit Ctrl+Enter. In the console pane the command **1:5** is run and the result – a vector of integers from 1 to 5 – is displayed beneath. The command lines have a prompt string **>** and the result line, in this case, starts with [1]. This is a guide that becomes useful when the output spans multiple lines. Try changing the command to **1:100*2**, either in the source view or directly in the console (you can hit the up arrow key to retrieve historical commands). Notice that the results now spread over multiple lines with the numbers at the left indicating the index number of the first element on that row. You can see that arrays indices in R start from 1 (not 0).

Let's bring in the other panes. To do this we need to declare some variables. Add the following code:

```
x <- 1:10
y <- rnorm(10)
d <- data.frame(x,y)
```

This assigns the range from 1 to 10 to the variable named **x**. Into the **y** variable we've inserted 10 random (normally distributed) numbers. Finally we create a data frame with these two variables as columns. Data frames are the fundamental data structure in R. To see the contents of **d** simply call it alone on the console or use **View(d)** to get a spreadsheet representation.

In the top-right pane you'll see the environment view. This is a list of the currently declared variables (**ls()**) alongside a summary of their contents that

Getting help

To get documentation on a given function in R you can call for **help**. To find out about the parameters and return value of the linear model function, for example, call **help("lm")**. You can also use the shorthand **?lm**, although this won't work with control syntax, meaning you have to call **help("for")**, for example. In *RStudio*, the F1 key will bring up help for the function under the cursor.

If you don't already know the name of the function you can use **help.search("distribution")** or **??distribution** to search the documentation. Sometimes you'll want to learn by exploring rather than having a particular goal in mind. You can browse package documentation with **help(package="ggplot2")**.

In addition to the basic API documentation, some packages also provide vignettes, which are PDF or HTML documents that provide an overview or some demonstrations. You can find them with **browseVignettes(package="colorspace")**. There are also demos, which you can find with **demo()** and run with eg **demo(graphics)**`.

has been generated by the **str** command, which is designed to give a reasonable, compact description of the structure of any object.

If we now execute the command **plot(d)**, the bottom-right pane will change to the plot view and you'll see a basic dot-plot of the random numbers.

Let's crank out some stats!

To run this shell command on different repositories we can create a function that will take a path and return a vector of file names. We will use interpolation to feed the path into a string then call this using **system**. The **paste()** command takes any number of arguments, converts them to character vectors, and concatenates them interleaving a separator – this defaults to the space character (if you don't want spaces you can use **paste(..., sep="")** or just **paste0(...)** for short. This function returns a list of files in the repository (in all sub-directories):

```
enumerate_files_in_repo <- function(repo_path) {
 command = paste("cd", repo_path, "; git ls-tree HEAD -r | awk -F '\t' '{print $2}'")
 system(, intern=T)
}
```

Once we have enumerated a list of files, we want to figure out who committed (ie "knows about") each line. We can use **git blame** with the **--line-porcelain** parameter to get a machine readable attribution for each line. We're only interested in the lines beginning with **author**, so we pipe the output via the **grep '^author '** command, and for each of these we only care about the bit after the field name, so we then pipe this via **sed -e 's/^author //'**. This gives us a long list of author names, one for each line in the source file.

```
get_author_for_lines_in_file <- function(repo_path, filename) {
 command = paste("cd", repo_path, "; git blame --line-porcelain", filename, "| grep '^author ' | sed -e 's/^author //'")
 system(command, intern=T)
}
```

If you call, for example, **author <- get_author_for_lines_in_file("/path/to/repo", "README")** then you'll get a vector of strings with the author of each line.

As you might expect from a statistical computing environment, there are quite a few different ways to count things. Here we will use **table** providing a "responseName" (rather than having the default column name "Freq") and we'll coerce the output to a data frame so we can combine results more easily.

```
count_line_authors <- function(author) {
 as.data.frame(table(author), responseName="line_count")
}
```

We should also keep a track of which file we used to derive these counts. We can do this by adding a column to the data frame with the filename:

```
file_blame <- count_line_authors(line_authors)
file_blame$filename <- rep(filename,nrow(file_blame))
```

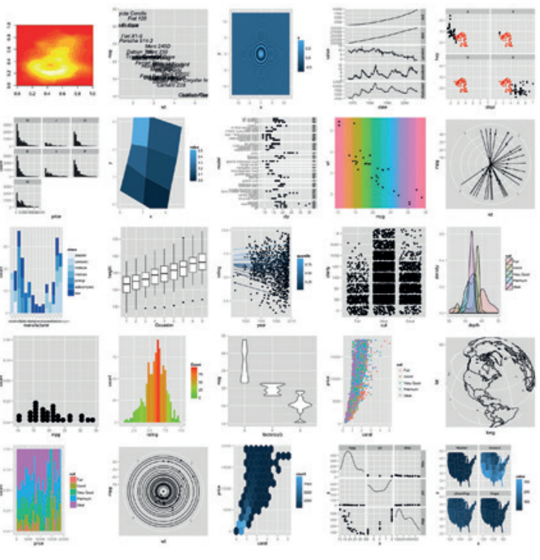
Encapsulating this in a function we get:

```
count_of_lines_by_author_in_file <- function(repo_path, filename) {
 line_authors <- get_author_for_lines_in_file(repo_path, filename)
 file_blame <- count_line_authors(line_authors)
 file_blame$filename <- rep(filename,nrow(file_blame))
 file_blame
}
```

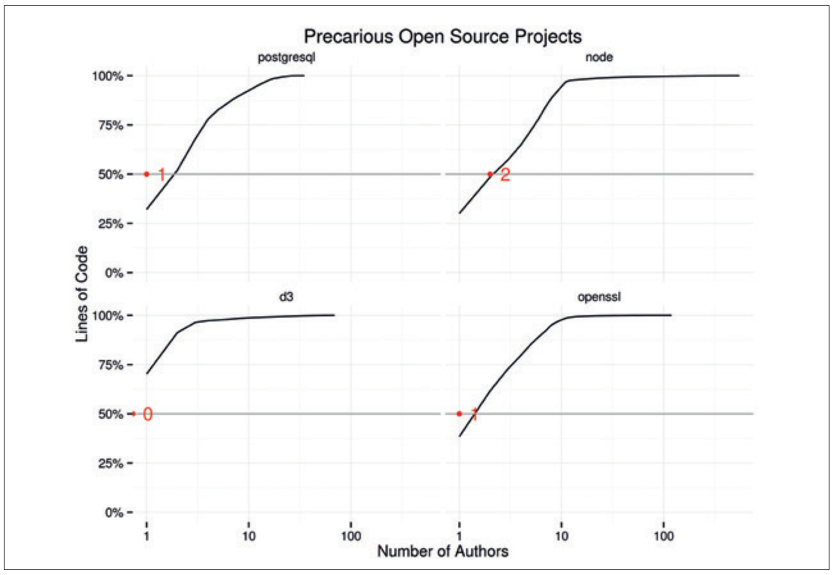
Now we need to call this function on each file and sum the results. Although R does have **for** loops, the idiomatic way of iterating like this is to use an **apply** function. The base R system comes with a handful of functions for this purpose, but instead we're going to use the **plyr** package, because it provides a consistent and intuitive interface. You can install this package with **install.packages("plyr")** and then load it with **library(plyr)**. The library follows a pattern described by

PRO TIP

R recognises arguments in function calls by either position or name. Here the **system** function has several default arguments, you can override a given parameter by name without having to specify all of the others (as you would need to do with positional arguments).



ggplot2 is a powerful visualisation library for R that implements the "Grammar of Graphics".



The bus factors of a handful of open source projects (higher is better/more robust). Low bus factors could be a sign that there aren't enough developers to spot security bugs in critical projects like *OpenSSL*.

the author, Hadley Wickham, as split-apply-combine: take something, split it into partitions according to some feature, apply a function to each piece, then combine the results.

We'll use the **adply** function as we want to take an array of filenames and turn it into a data frame of line counts by author. The first argument is our list of filenames; **adply** also needs to know how to split this up. In our case this second argument is simply "1" (ie "by row"), the third argument is the function that will be applied to each entry in the split-up array (**count_of_lines_by_author_in_file**). Since we also need to specify the **repo_path** we add this as a named argument. The **git blame** command can take a non-negligible amount of time to run, and it's being run many times, so to have some feedback during the process we provide the final argument to get a progress bar.

```
lines_by_file <- adply(repo_tree, 1, count_
of_lines_by_author_in_file, repo_
path=repo_path, .progress = "text")
```

PRO TIP

There's more than one way to skin an analytical cat so it's important to explain your methodology and the implications it has for your conclusions. The beauty of writing and sharing R code is that your research can be reproduced and validated by others.

This gives us a data frame with a row for each file and author (many authors will appear multiple times as they contribute to different files). To calculate a grand total across all the files in the repository we use another **plyr** function, this time from one data frame to another: **ddply**. The signature for this function is similar to **adply**, but we specify the split with column names, not an integer, here provided with a quote function **.(author)** that prevents **author** from being eagerly evaluated when the function is called, instead evaluating it in the context of the data frame (as a column name). Again we pass a function to apply; in this case we use **summarise**, which also expects pairs of **var=value** to describe variables that it should add to the result. Here we want to sum the **line_count**s:

```
ddply(lines_by_file, .(author), summarise, line_
count=sum(line_count))
```

We can tie all this together with another function:

```
count_of_lines_by_author_in_repo <- function(repo_
path) {
 repo_tree <- enumerate_files_in_repo(repo_path)
 lines_by_file <- adply(repo_tree, 1, count_of_lines_by_
author_in_file, repo_path=repo_path, .progress = "text")
 ddply(lines_by_file, .(author), summarise, line_
count=sum(line_count))
}
```

We now have a data frame with two columns: **authors** and **line_count**. You can see from this which authors have made the biggest contribution (and hold the most knowledge) by sorting the data frame:

```
author_lines[order(author_lines$line_count,
decreasing=T),]
```

The **order** function returns a vector of indices that we use to extract rows from the data frame in the order we wish. If you have a very large list of contributors you might want to wrap this with the **head** function.

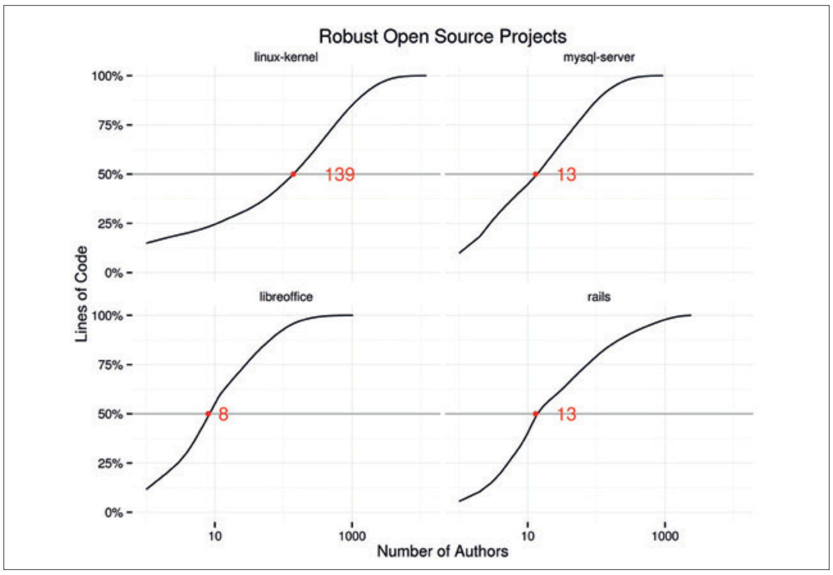
I am become death, destroyer of developers

If you'll forgive a morbid extension of our metaphor, this is a bit like lining up our authors in front of the bus with the biggest contributor first. We need to know how far the bus has to travel before the project is in serious trouble. We're less interested in knowing who they are, and instead want to know how many we can afford to lose (in the worst case scenario). We also need to know what we're losing in terms of knowledge as the bus makes its grisly journey. In analytical terms this means we'll sort the authors in order of contribution, then calculate the cumulative number of authors and lines of code as we add (or remove) additional contributors. Since we want to compare this project to others we also want to calculate percentages for this variables. The following function wraps up these details – note that before we sort the data frame, we're also sorting the **authors** dimension (which will be useful if you want to plot this):

```
calculate_author_contribution <- function(author_lines) {
 author_lines$author <- reorder(as.factor(author_
lines$author), -author_lines$line_count)
 sorted_contributions <- author_lines[order(author_
lines$line_count, decreasing=T),]
 sorted_contributions$cumulative_line_count <-
cumsum(sorted_contributions$line_count)
 sorted_contributions$cumulative_author_count <-
1:nrow(sorted_contributions)
 sorted_contributions$cumulative_line_percent <-
sorted_contributions$cumulative_line_count/
max(sorted_contributions$cumulative_line_count)
 sorted_contributions$cumulative_author_percent
<- sorted_contributions$cumulative_author_count/
max(sorted_contributions$cumulative_author_count)
 return(sorted_contributions) }
```

Before we can calculate the bus factor, we must make a decision: how much knowledge do we need to lose before we would say the project is in serious trouble? For simplicity's sake we'll say half (the charts show that this is a fair assumption for most projects).

To see 50% of the **line_count**:



Projects with larger teams can better afford to lose individual developers and have higher bus factors overall.

```
calculate_bus_factor <- function(author_contribution,
critical_threshold=0.5) {
 critical_contributions <- author_contribution[author_
contribution$cumulative_line_percent < critical_
threshold, ]
 nrow(critical_contributions)
}
```

So, now that we have an algorithm, let's calculate the bus factor for some open source projects. Furthermore we can also visualise the relationship between number of people and proportion of the code base rather than simply looking at the 50% mark.

We've gathered data from a range of open source projects and bound these **author_contribution** data frames together into a single data frame with an additional **repo** column that distinguishes the project (for brevity we've omitted the code here, but you can find it on GitHub). To visualise the data we first need to load two libraries:

```
library(ggplot2)
library(scales)
```

We also need a function to calculate the bus factor for a vector of **cumulative_line_percent** values:

```
bus_factor_for_line_percent <- function(lp, critical_line_
percent=0.5) {
 length(lp[lp<critical_line_percent])
}
```

Then we calculate the bus factors for each projects:

```
bus_factors <- ddply(all_ac, .(repo), summarise, bus_
factor=bus_factor_for_line_percent(cumulative_line_
percent), critical_line_percent=0.5)
```

Finally we can plot a graphic. We'll step through this line by line (the **+** is used to compose plots with **ggplot**; we have it on the end of the line to let the interpreter know we're not done with the specification). The **ggplot** function expects a data frame, then an aesthetic mapping specified with the **aes** function. We want to plot the number of authors against the proportion of the code base. Tshis call to aes maps the **cumulative_author_count** variable to the "x" axis and **cumulative_line_percent** to the "y":

```
ggplot(all_ac, aes(cumulative_author_count,
cumulative_line_percent)) +
```

Add a **geom** to determine how the data will be displayed on the screen. Here we choose a simple line to connect the data points:

```
 geom_line() +
```

Expand the limits of the graph so that the origin of 0,0 is in view (ie no authors, no lines):

```
 expand_limits(y=0, x=0) +
```

Specify the scales. In the case of the x axis, we want a logarithmic transformation, as the interesting bit is first few authors. We also add titles, and set the y axis to use percentages for labels (10% rather than 0.1):

```
 scale_x_log10("Number of Authors") + scale_y_
continuous("Lines of Code", labels=percent) +
```

Draw a horizontal line at 50% to remind us of the critical threshold:

```
 geom_hline(y=0.5, colour="darkgrey") +
```

Divide the chart up into separate versions (or facets) one for each repository:

```
 facet_wrap(~ repo) +
```

Annotate each facet with a red point and a label to highlight the bus factor:

```
 geom_point(data=bus_factors, aes(bus_factor, critical_
line_percent), colour="red") + geom_text(data=bus_
factors, colour="red", aes(bus_factor, critical_line_
percent,label=bus_factor), hjust=-1) +
```

Provide an overall title:

```
 labs(title="Bus Factor for Open Source Projects") +
```

Then finally set the theme to minimal (this is a personal preference – check out the **xkcd** package for a familiar-looking alternative):

```
 theme_minimal()
```

You can see in the images on these pages that some projects are quite safe. The Linux Kernel, for example, should continue on if the worst happens, while others such as *OpenSSL* are a little more precarious. So do you fare any better?

Robin Gower became interested in the statistical language of GNU-R after looking for languages suitable for pirates. Ar.

RUST: GET STARTED WITH MOZILLA'S NEW LANGUAGE

Mozilla may be famous for Firefox, but it's also given us a great language to play with.

MIKE SAUNDERS

WHY DO THIS?

- Expand your skills with a new programming language
- Discover why Rust is the hot topic among coders
- Write memory-safe and highly concurrent software

Does the world really need another programming language? We already have C, C++, C#, Objective C, Swift, Java, Perl, Python, Ruby, Go and hundreds of lesser-known languages with just a handful of users. Surely that's enough for everyone, right? Well, the Mozilla team doesn't think so, and in the last few years has been working on Rust, a "systems programming language that runs blazingly fast, prevents nearly all segfaults, and guarantees thread safety". In other words: it lets you write low-level software with C-like performance, but without the problems that plague that language.

Rust is designed to be multi-paradigm, in that it can be used in procedural, functional and object-oriented programming. It doesn't allow the use of null or dangling pointers, making it much more memory safe than C (see Core Tech on page 94 for more on why these are bad), and provides a set of streamlined runtime libraries handling I/O, concurrent programming and other features. Rust is a relatively new language, with its 1.0 release only arriving in May 2015, but it has been in development for over five years. And it's well worth learning it now – at least the basics – as it could prove to be a major competitor for C and C++ in the coming years.

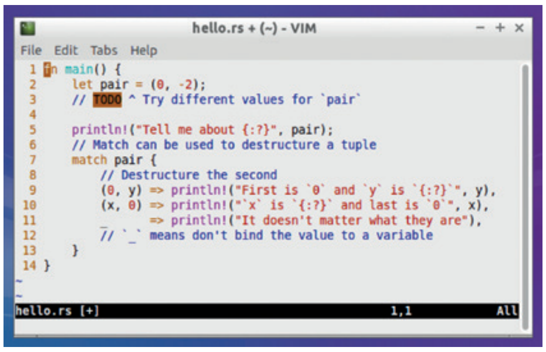
Rust may be available in your distro's package repositories, but as the language is undergoing development it's worth getting the latest binaries from the project's website at **www.rust-lang.org**. Grab the **rustup.sh** script and run it, eg:

```
wget -O - https://static.rust-lang.org/rustup.sh | sh
```

(This uses **wget** to retrieve **rustup.sh**, spit it to **stdout**, and pipe the results to a shell to execute it.) After a few minutes of downloading, Rust will be installed and you can start compiling programs using the **rustc** command. Let's kick off with a classic – type this in and save it as hello.rs:

> Rust is designed to be multi-paradigm, in that it can be used in procedural, functional and OO programming.

```
// Hello world
fn main () {
        println!("Hello world");
}
```



```
fn main() {
    let pair = (0, -2);
    // TODO ^ Try different values for `pair`

    println!("Tell me about {:?}", pair);
    // Match can be used to destructure a tuple
    match pair {
        // Destructure the second
        (0, y) => println!("First is `0` and `y` is `{:?}`", y),
        (x, 0) => println!("`x` is `{:?}` and last is `0`", x),
        _      => println!("It doesn't matter what they are"),
        // `_` means don't bind the value to a variable
    }
}
~
~
hello.rs [+]                                  1,1            All
```

Many editors support syntax highlighting for Rust, such as *Vim* (see **https://github.com/rust-lang/rust.vim**).

You'll notice straight away that the syntax is rather like C/C++, with double slashes preceding comments, code blocks surrounded by curly brackets, and parentheses used for the arguments in a function definition. Note here that functions are defined with the **fn** keyword, and every program must include a **main()** function. The exclamation mark after **println** here indicates that it's a macro (it provides a friendly wrapper around the Rust runtime's own print routine).

To compile it, simply enter:

```
rustc hello.rs
```

You'll see a resulting binary, **hello**, which you can execute using **./hello**. But if you look at the size of the binary, you may be rather shocked: it's over 800kB! All that for such a simple program? Well, by default the Rust compiler statically links in a good chunk of the language's runtime, so that you can copy the binary to an installation without Rust installed and run it without problems. You can force the compiler to use optimisations and dynamic linking, however:

```
rustc -O C prefer-dynamic hello.rs
```

Now the binary is a much more respectable 8kB in size, but if you run **ldd** on it you'll see that it needs **libstd-<version>.so** to be installed.

Language syntax

Now that we can compile and execute Rust programs, let's look at the language's syntax and see how it differs from C, C++ & co:

```
fn doubler (x: i32) -> i32 {
        x * 2
}

fn main () {
        let a: i32 = 5;
        let b;

        b = doubler(a);
        println!("a doubled is {}", b);

        match b {
                1 ... 10 => println!("From 1 to 10"),
                _ => println!("Another number"),
        }
}
```

If you come from a C/C++ background, you may think this looks rather messy, but there's logic behind it. Let's kick off with the **main()** function: in the first **let** line, we declare a variable, **a**, to be a 32-bit signed integer containing the value 5. We can omit the variable type (i32 is the default) and the initial value, in which case it will be zero. Note that if you declare a variable and assign it a specific value like we do with **a** here, you can't change it later, so this would generate an error:

```
let a: i32 = 5;
a = 10;
```

By default, variables in Rust are not mutable – that is, they can't be changed. You have to explicitly declare them as mutable like so:

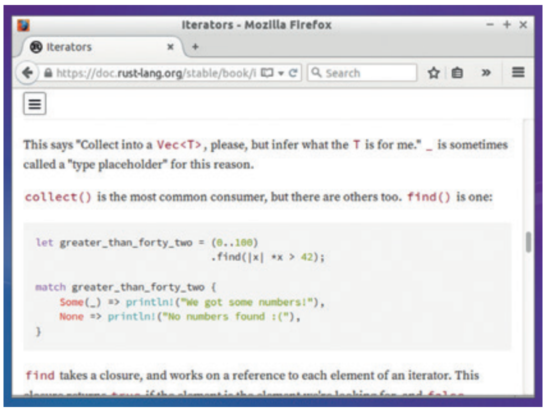
```
let mut a: i32 = 5;
```

What's the point of this? Isn't it just extra typing work? Well, yes, but it's all about writing safe programs. You should only make things mutable that absolutely have to be changed. Rust encourages you to be verbose, but it's all for clarity: in the above line, we know that **a** is a signed integer of exactly 32 bits, with permission to change its value in the future.

Next, we call our **doubler** function with **a** as the calling value, and **b** to store the result. Note the function definition for **doubler** at the top of the code: the type of the passed parameters is specified (an i32), and the return value type is given after the **->** characters. You'll also notice that the only code we have in the function is **x * 2**, which isn't even followed by a semi-colon as in normal Rust code – so what's going on here?

It's perfectly possible to return a value in normal C-like fashion, but you can also achieve the same result by providing an expression as the last line in the function, as we do here. And because it's just an expression, it doesn't need the semi-colon to terminate it.

Back in the **main()** function, we use the **println!()** macro to print the result; note the variable substitution in the **{}** curly brackets. Finally, we demonstrate Rust's very useful "**match**" keyword, which saves a lot of typing if you need to perform a lot of if/else operations. Here, **1 ... 10** refers to a range (numbers between 1 to 10 inclusive), while the underscore (**_**) refers to everything else.


Iterators - Mozilla Firefox
Iterators
https://doc.rust-lang.org/stable/book/i
Search

This says "Collect into a Vec<T>, please, but infer what the T is for me." _ is sometimes called a "type placeholder" for this reason.

collect() is the most common consumer, but there are others too. find() is one:

```
let greater_than_forty_two = (0..100)
                             .find(|x| *x > 42);

match greater_than_forty_two {
    Some(_) => println!("We got some numbers!"),
    None => println!("No numbers found :("),
}
```

find takes a closure, and works on a reference to each element of an iterator. This

Rust's documentation is detailed and well written – a major selling point for a fairly new language.

In Rust, the **char** variable type is actually four bytes and can contain any Unicode value, so the language is designed out-of-the-box to handled a wide range of writing scripts and special characters. Another useful type is the tuple, which is a list of variables with different types:

```
let x = (1, 2.0, "Hello");
```

Here we have an integer, a floating point number, and a string – all in the same tuple. These elements are immutable, and we can access them like so:

```
println!("{}", x.2);
```

This prints the third element in the **x** tuple, so **Hello**.As in traditional arrays, which Rust also supports, indexing begins with zero. You can use tuples to return multiple values from a function:

```
fn switch(input: (i32, i32)) -> (i32, i32) {
        (input.1, input.0)
}

fn main() {
        let x = (10, 50);
        let y = switch(x);
        println!("{}, {}", y.0, y.1);
}
```

Here we have a function called **switch()**, which takes a tuple of two i32 values and stores them in the variable **input**. It also returns a tuple of two i32s. Inside this function we have a simple expression, a tuple with the items back-to-front – that's what's returned to the calling code.

In our **main()** function, we create a tuple called **x** containing the values 10 and 50, and a tuple called **y**, which contains the results after **switch()** is called. Then we print the results to the screen (**50, 10**).

So that's a brief introduction to the syntax and features of Rust – if you'd like us to go more in-depth with a tutorial series, let us know!

PRO TIP

If you're ready to explore more of Rust's features on your own, check out the official documentation at **https://doc.rust-lang.org/book**.

Mike Saunders codes to the baroque rhythms of Bach, Buxtehude and the original Gameboy's Tetris music.

ADA: MILITARY-GRADE PROGRAMMING

Rock solid reliabilty, a solid safety record in embedded systems, and a cool name.

JULIET KEMP

In the 1970s, the US Department of Defense (DoD) had lots of embedded computer system projects, and over 2,000 languages being used for them. The languages were often either obsolete or hardware-dependent, and none of them were safely modular. There were obvious concerns about safety and long-term usability/maintenance (finding maintainers for 2,000 obsolete languages is a tough gig), not to mention the possibility of having to scrap perfectly good hardware because the software couldn't be safely altered. In 1975, they created a working group to find or create a suitable programming language to standardise on for themselves and for the UK Ministry of Defence, with the intention of solving the 'software crisis'.

In the customary manner of government projects, committees, and computer language projects, it took a while. In fact, it took three years and a number of drafts before they had even reached Steelman language requirements for their proposed language; but after those three years, they were also able to conclude that there was no existing language that met them. They wanted a focus on reliability, maintenance, and efficiency, aimed as the language would be at embedded systems; and they wanted a language that could be used for any of the systems they needed. The ideal (which they never quite reached) would be a single language across every DoD computer project there was.

Four contractors were hired to develop proposals, one of which would be chosen to develop fully. Interestingly, all of the four proposals had a basis in Pascal, the Algol-like language developed in the late 1960s as a teaching language. In 1979, the Green team proposal was chosen and named Ada, after Ada Lovelace. It had been developed by a French team at CII Honeywell Bull, led by Jean Ichbiah.

Ada 83 was structured and strongly typed, with built-in support for concurrency, run-time checking, parallel processing, and exception handling. It also provided task/parallel processing support, with the rendezvous mechanism allowing tasks to synchronise with one another. Real-time support was part of the tasking features. As it was designed for developing large systems, it was also highly modular, and very readable and maintainable if written properly.

The Ada Language Mascot, by Leah Goodreau, inspired by Babbage's nickname for Lovelace, "Lady Fairy".

Ada was very fully-featured, but that meant that it was also quite complex, and early compilers tended to be slow. The language validation suite that was mandated for Ada compilers made for a very reliable language, which was the aim; but it may also have tended to make the compilers slower, or at least slow down their improvement, as the focus when developing compilers was on passing the validation suite rather than on speeding up compilation or run-time performance. On the other hand, compiler validation, and the numerous compile-time checks that compilers were mandated to perform, ensured (or tried to) accuracy and reliability, clearly important in safety-critical systems. But slow compilation and poor run-time performance can also have maintenance and safety implications, making it slow to improve and debug systems, so it's not a straightforward trade.

Ada's focus on safety-critical support has made it popular for other projects with low risk tolerance. Although it's less used in military situations now, it's still widespread in transport (air and rail, including UK trains and air traffic control), space software, banking, and medical applications.

Getting started

Gnat is the best Ada compiler for Linux, and should be available via your package manager. Once you've installed it, create a file **hello.adb**:

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
  Put_Line ("Hello World");
end Hello;
```

Ada is designed to have a long life, and to be maintained by people other than the original programmer, so it's considered vital to make code as readable as possible. (Anyone who's ever been faced with an aging and incomprehensible code base will understand the value of this.) So the code should be pretty easy to read through.

Ada.Text_IO is a standard package, which deals (as expected) with text input/output. **with** imports the package, and **use** is one of the ways to make the package visible, enabling you to use procedures (like **Put_Line**) from the package. Ada was designed to be highly modular and to encourage modular programming, so packages are a major feature.

The main procedure can be called whatever you like, but it's good practice to match the filename. Semi-colons are used to end statements, and also to end a procedure. Compile the file with **gnatmake hello.adb**, and run it with **./hello**.

Tracking time

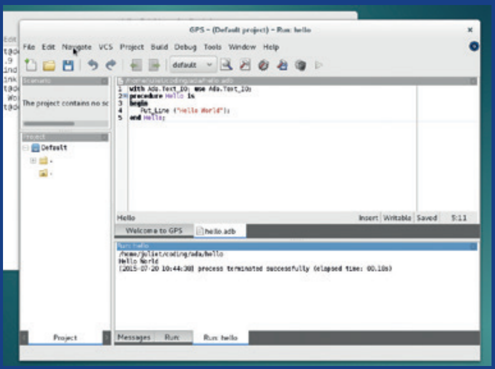
For something a bit more involved, let's try writing a program to track how much time you've spent on a particular task. Set up a new project in *GPS* (the default options should all be fine), and open a new file called tracking.adb:

Building and running within GPS

If you want an Ada IDE, you can open up *GPS* (**gnat_gps**), which is used for the tutorial screenshots.

To build a file within *GPS*, first you need to add it to the project file. Go to Project – Edit Project Properties, and choose the Main Files tab. Click add, choose the filename (eg **hello.adb**), and click OK. Once the file is added, go to Build > Project > **hello.adb**. You'll get a build window, but as a rule the default options are fine.

You can then go to Build > Run > hello, and run it either in the GPS Run window (as in the screenshot below) or in an external terminal.


GPS – (Default project) – Run: hello
File Edit Navigate VCS Project Build Debug Tools Window Help
default
Scenario
The project contains no sc
Project
Default
Hello
Insert Writable Saved 5:11
Welcome to GPS
hello.adb
Run: hello
Project
Messages Run Run: hello

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
   Put_Line ("Hello World");
end Hello;
```

```
/home/juliet/coding/ada/hello
Hello World
[2015-07-20 10:44:38] process terminated successfully (elapsed time: 00.10s)
```

Building and running a file within GPS.

```
with Ada.Calendar, Ada.Calendar.Formatting, Ada.Text_IO,
  Ada.Strings.Unbounded, Ada.Strings.Unbounded.Text_IO;
use Ada.Calendar, Ada.Calendar.Formatting, Ada.Text_IO,
  Ada.Strings.Unbounded, Ada.Strings.Unbounded.Text_IO;

procedure Tracking is
 track : Unbounded_String;
 date : Time := Clock;
begin
 Put_Line("What are you tracking?");
 track := Get_Line;
 Put_Line(track);
 Put_Line(Image(Date => date));
 New_Line;
end Tracking;
```

You'll notice that we've imported a whole raft of libraries this time, mostly to do with unbounded strings and the calendar.

Normal Ada strings have to be exactly the length that they are specified to be. So if you put in too many characters, you'll lose some; if you put in too few characters, your program will halt, waiting for the string to become full. The standard libraries **Ada.Strings.Bounded** and **Ada.Strings.Unbounded** solve this problem. Bounded strings have a maximum size, and are a bit faster, whereas unbounded strings have no maximum size and are much more flexible, but because they're dynamically allocated are a bit slower. We'll use unbounded strings for their flexibility.

So at the top of the procedure, we declare two variables, an unbounded string, and a time value. We also set date, the time value, to the current system clock time, thus declaring it (with **:** and assigning it (**with :=**) in a single line. In the body of the procedure, we output a line and use **Get_Line** to get the unbounded string input by the user. (The regular **String** equivalents are **Put()** and **Get()**.)

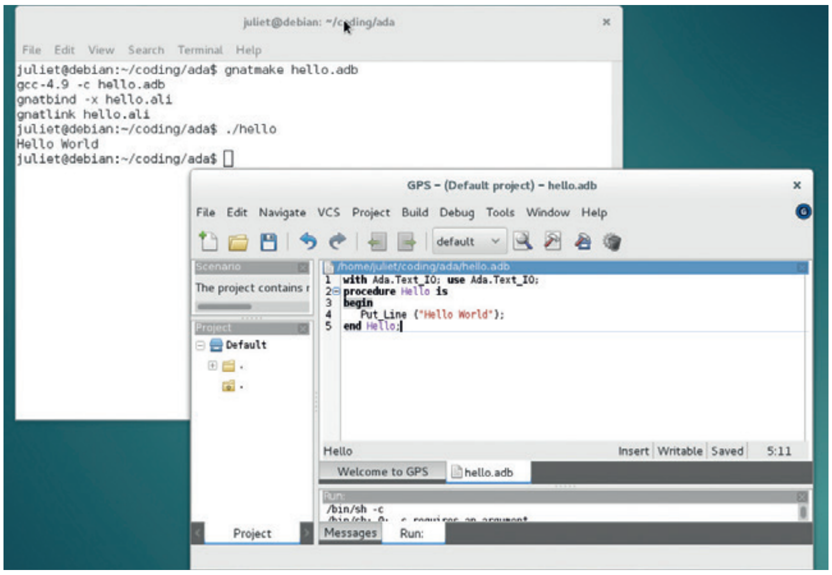
Finally we output the input string and the current time to the screen. **Image** is an Ada attribute which can be used to provide a string representation of its input, which is handy in cases like this. We'll use more attributes later on.

However, it would make more sense to store our tracking data – string, date, and hours completed – in a single entity. The Ada Record composite type, which lets you define your own records, is useful for this:

```
procedure Tracking is
 type Track_Record is
  record
   Name : Unbounded_String;
   Date : Time;
   Hours : Integer;
  end record;
```

PRO TIP

Note that if you need to refer to Procedure A in Procedure B, Procedure A must be declared before Procedure B, or *GNAT* won't be able to find it.


juliet@debian: ~/coding/ada
File Edit View Search Terminal Help
GPS – (Default project) – hello.adb
File Edit Navigate VCS Project Build Debug Tools Window Help
default
Scenario
The project contains r
Project
Default
/home/juliet/coding/ada/hello.adb
Hello
Insert Writable Saved 5:11
Welcome to GPS
hello.adb
Run:
Project
Messages
Run:

```
juliet@debian:~/coding/ada$ gnatmake hello.adb
gcc-4.9 -c hello.adb
gnatbind -x hello.ali
gnatlink hello.ali
juliet@debian:~/coding/ada$ ./hello
Hello World
juliet@debian:~/coding/ada$ 
```

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
   Put_Line ("Hello World");
end Hello;
```

```
/bin/sh -c
```

Hello World edited in the IDE and running on the command line.

```
 track : Track_Record;

 procedure Output_Track_Record(track : in Track_
Record) is
 begin
  Put_Line(track.Name);
  Put_Line(Image(Date => track.Date));
  Put_Line(Item => Integer'Image(track.Hours));
  New_Line;
 end Output_Track_Record;

begin
 Put_Line("What are you tracking?");
 track.Name := Get_Line;
 track.Date := Clock;
 Put_Line("How many hours have you done?");
 track.Hours := Integer'Value(Get_Line);
 Output_Track_Record(track);
end Tracking;
```

The first section just defines a new record type, with a name, the date of the last update, and the hours spent on this particular thing. The next section defines a sub-procedure. Ada has a couple of ways in which you can do this; see the boxout for more information. Here, the procedure has a single input (**in**) value, named **track** and with type **Track_Record**.

Most of the rest of this will be familiar. The **Integer'Image** and **Integer'Value** are ways of translating from integer to string, and back, using attributes. **'Value** (you can also use **Float'Value**, etc) gives the scalar value of a string, and **'Image** gives a string representation of a scalar. Attributes in general are things that can apply to different types of objects.

Output to file

So far so good, but at present we don't have any way of saving this information. To fix that, we could take the record apart, write it to a text file, and then reconstruct it when reading in. However, a better (and neater) bet is to write it out as a binary record object:

```
with Ada.Directories, Ada.Streams.Stream_IO;
use Ada.Directories, Ada.Streams.Stream_IO;
procedure Tracking is
 -- ... as before ...
 filehandle : Ada.Streams.Stream_IO.File_Type;
 fileaccess : Ada.Streams.Stream_IO.Stream_Access;

 procedure Save_Records_To_File(File : in out Ada.
Streams.Stream_IO.File_Type;
                 Name : in String) is
 begin
  Create(File, Out_File, Name);
  fileaccess := Stream(File);
  Track_Record'Write(fileaccess, track);
  Close(File);
 end Save_Records_To_File;

begin
 -- as before
 Save_Records_To_File(filehandle, "trackingdb");
end Tracking;
```

There are several ways in which you can write out binary data in Ada, but the most flexible is **Stream_IO**, which also works with **Unbounded_String**, so we'll use it here.

To set up, we need a **filehandle** variable and an access stream through which we'll access the data once the file is open. We then create a new sub-procedure, which has two arguments. **File** is the filehandle, and **Name** is the string name of the file. We create a new file, define **fileaccess** as the access stream of that file, then write the record using the **'Write** attribute of the **Track_Record** type. This will convert the **Track_Record** object into something that can be written to the stream, given the stream and the object to write as arguments. Close the file again, and we're done. Remember to add a line to the main procedure which calls **Save_Records_To_File**.

What we can't do yet is to read the data back out again, or to append new data to an existing file; **Create** will overwrite the file each time. So let's add a couple of functions to fix these problems:

```
with Ada.Directories, Ada.Streams.Stream_IO;
use Ada.Directories, Ada.Streams.Stream_IO;
procedure Tracking is
  type Track_Record is -- as before

  type Record_Array is array(1..100) of Track_Record;
  records : Record_Array := Record_Array'(others =>
                      (Hours => -1,
                       Name => To_Unbounded_
String(""),
                       Date => Clock));
  counter : Natural := 1;
  track_file : String := "trackingdb";
  -- other variables as before
```

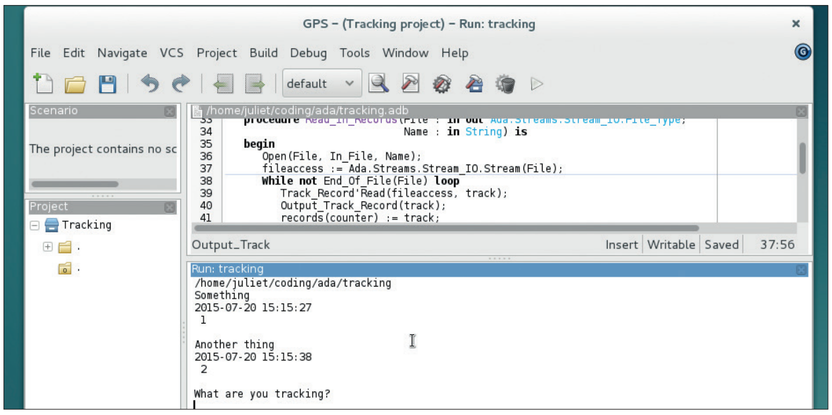
First up, we'll need a structure to read our data into. Ada arrays are very powerful, but they're not dynamic: the array is of a fixed size. Array elements can be of any type, but they must all be the same type; and array indexes can be of any type, not just numbers.

Subprograms: procedures and functions

Ada has two types of sub-programs. Procedures have no return value, whereas functions do return a value. The basic structure of both procedures and functions is exactly the same as of the main procedure, with declarations at the start and then a **begin...end** statement block.

Procedures and functions have three modes for their input variables:

- **in** means that the parameter will not be changed. It is treated as a constant within the procedure.
- **in out** means that the parameter may be changed. It is treated as a variable within the procedure and any changes will be permanent.
- **out** means that the parameter will be given a value during the procedure, and its previous value won't matter. This gives you a way of 'returning' a value from a procedure. Look for these in the various procedures and functions in the tutorial.


```
GPS – (Tracking project) – Run: tracking
/home/juliet/coding/ada/tracking.adb
   procedure Read_In_Records(File : in out Ada.Streams.Stream_IO.File_Type;
                             Name : in String) is
   begin
      Open(File, In_File, Name);
      fileaccess := Ada.Streams.Stream_IO.Stream(File);
      While not End_Of_File(File) loop
         Track_Record'Read(fileaccess, track);
         Output_Track_Record(track);
         records(counter) := track;
Output_Track                      Insert  Writable  Saved   37:56
Run: tracking
/home/juliet/coding/ada/tracking
Something
2015-07-20 15:15:27
 1

Another thing
2015-07-20 15:15:38
 2

What are you tracking?
```
Successfully running! Two items stored, asking for another input.

We'll use numbers here, though.

When initialising an Ada array, you can initialise specific members of the array. You can also (since Ada 95) use the **others** keyword to initialise the rest of the array. Here, as there's no separate initialisation values, **others** applies to the whole thing. If you don't do this, you don't know what will go into the array before you write into it.

Next, reading in the records:

```
procedure Read_In_Records(File : in out Ada.Streams.
Stream_IO.File_Type;
            Name : in String) is
begin
 Open(File, In_File, Name);
 fileaccess := Ada.Streams.Stream_IO.Stream(File);
 While not End_Of_File(File) loop
  Track_Record'Read(fileaccess, track);
  Output_Track_Record(track);
  records(counter) := track;
  counter := counter +1 ;
 end loop;
 Close(File);
end Read_In_Records;
```

It's structured much like our previous output procedure; take a filehandle and a filename, open the file, and create an access stream. You can use **While...loop...end loop**; to iterate over a filestream, not just an array or similar structure, and it's a neat way to do it, especially with the **End_Of_File** function. We use the **Read** attribute to read in records, output each one to the screen, store it in the array, and increment the counter. Don't forget to close the file! Here's the last bit:

```
 procedure Get_New_Track is
 begin
  Put_Line ("What are you tracking?");
  -- Get name, date, and hours as in main procedure
earlier
  records(counter) := track;
 end Get_New_Track;

 procedure Save_Records_To_File(File: in out Ada.
Streams.Stream_IO.File_Type;
            Name : in String) is
 begin
  Create(File, Out_File, Name);
  fileaccess := Stream(File);
  for T in records'Range loop
   if (records(T).Hours /= -1) then
    Track_Record'Write(fileaccess, records(T));
   end if;
  end loop;
  Close(File);
 end Save_Records_To_File;

begin
 if(Exists(track_file)) then
  Read_In_Records(filehandle, track_file);
 end if;
 Save_Records_To_File(filehandle, track_file);
end Tracking;
```

Get_New_Track is just breaking out the code that was in the main procedure, with the addition of a line to store the new record in the array. **Save_Records_To_File** is pretty similar to our previous single-record code, too; we just use a **for...loop...end loop**; structure to write out the whole array one at a time. We're still overwriting the file, but as we've already read all the records into the array, we're overwriting it with all the available data. (To improve performance, look into changing this to append only a single value.) The **'Range** attribute here does what you'd expect, providing the range of indices into the array (in our case, 1–100). You might have noticed by now that attributes are ubiquitous in Ada and are both neat and powerful. Note too the **if...then...end if;** structure. Finally, a couple of lines in the main procedure to kick the whole thing off.

Ada is a huge and fully-featured language, so there are many more advanced aspects to explore, such as the concurrency and other multithreading features, or the more advanced OO features. It's well-documented (as it's ISO-standard compliant), and there is still an active community, especially around Ada 2012. As the mascot logo says, it's time-tested, safe, and secure; the language for a complex world.

Juliet Kemp is a friendly polymath, and is the author of Apress's Linux System Administration Recipes.

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.

Garbage Collection

Learn how high-level languages like Python reclaim memory, so you can concentrate on your job and not memory management.

Nobody seems to enjoy cleaning up after themselves. If you have children, or remember yourself being a kid, you'd probably agree. Adults aren't too keen on housekeeping activities either, as robotic vacuum cleaners sales figures suggest. Programmers are also humans, so it's no wonder they prefer computers to keep the environment clean after their programs. This makes development easier, and hard-to-debug-yet-easy-to-exploit bugs less likely to occur. Of course, I'm speaking of automatic memory management and its thickest pillar – garbage collection.

Do it yourself

The C language tutorial we ran in LV018 dealt with how memory management should work. Unfortunately, things can go wrong. Take a look here:

```
char * get_name() { /* Implementation */ };
void somefunc()
{
  char *name = get_name();
  printf("Your name is %s\n", name);
}
```

The tricky part is, who should free the memory pointed by **name** when it's no longer needed? The correct answer is that it's up to the **get_name()** implementation. If it allocates memory, a caller should **free()** it. This may sound trivial, but real-world programs have more than two functions, and pointer ownership is rather difficult to track.

Okay, what about this code?

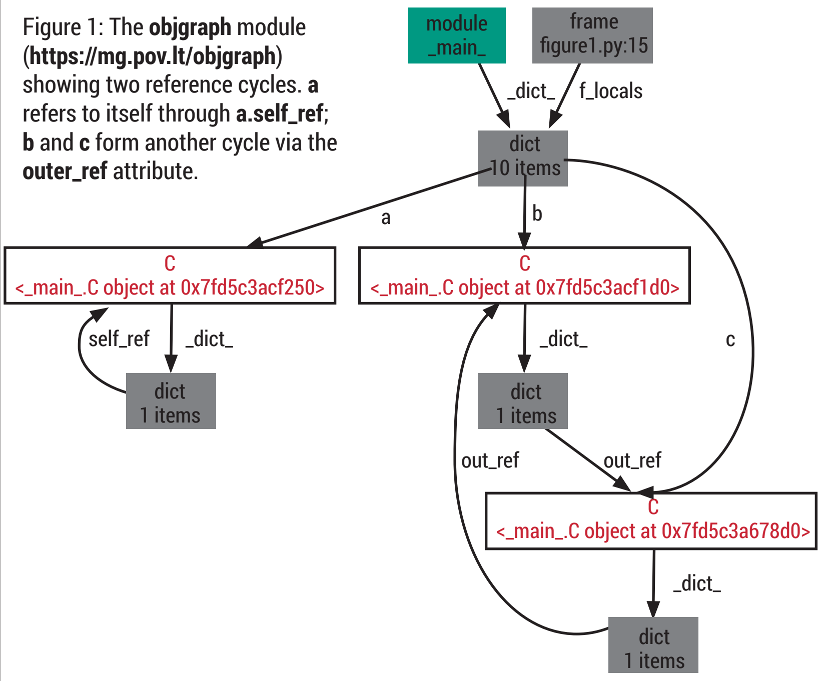
```
void print_name() { /* Implementation */ };
void somefunc()
{
  char *name = get_name();
  print_name(name);
  free(name);
}
```

It may work just fine. Or, crash randomly if **print_name()** remembers **name** to re-use it later. This situation is known as a "dangling pointer", and the problem is that the pointer's value doesn't say whether the object is still alive.

These are just two problems, but hopefully you've got their taste. And they aren't unsolvable: for example, some C++ libraries provide so called "guarded pointers" that can't dangle. Manual memory management is somewhat like driving a car with manual transmission: tedious, but fine if you know what you are doing. Pulling the lever randomly is almost certainly a way to get into trouble.

This analogy goes even further: just like automatic gear, automatic memory management comes with a performance hit. Quite often, it also introduces random pauses during the program's execution, which is bad for time-critical code. Still, automatic memory management makes programming easier

Figure 1: The **objgraph** module (**https://mg.pov.lt/objgraph**) showing two reference cycles. **a** refers to itself through **a.self_ref**; **b** and **c** form another cycle via the **outer_ref** attribute.



and less error-prone, which is usually enough to justify its costs. The majority of high-level languages feature automatic memory management. In this Core Tech, we'll stick to Python's implementation, as it is full-featured yet relatively simple.

Reference counting

The simplest form of automatic memory management is reference counting, which the Linux kernel uses to track which modules are safe to unload. The idea is simple: each object receives a counter. Code that creates a reference to the object increments it. When this reference vanishes, the counter is decremented. When it reaches zero, the object has no users and is safe to delete.

Reference counting is often available out of the box. For example, the C++ standard library includes smart pointer classes for this purpose. Python also uses reference counting to keep track of allocated objects:

```
>>> a = 1024
>>> b = a
```

The reference counter of **a** starts with 1. **b** adds another reference, as we can see:

```
>>> import sys
>>> sys.getrefcount(a)
3
```

This returns 3, not 2, because **getrefcount()** also references **a** through its argument. If you break the reference through **b**, the result will drop by one:

```
>>> b = None
>>> sys.getrefcount(a)
2
```

Note that executing **del b** has the same effect. Actually, **del** neither deletes an object nor calls its destructor. It just decrements reference counter and wipes the name:

```
>>> b = a
>>> del b
>>> sys.getrefcount(a)
2
>>> b
NameError: name 'b' is not defined
```

Nobody is perfect, and reference counting is no exception. At very minimum, you sacrifice the memory that the counters consume, though this isn't usually a big deal, as objects tend to be large. Keeping counters updated also adds overhead to common pointer operations; this can be optimised as well.

The real problem is reference cycles:

```
>>> a = list()
>>> a.append(a)
```

Here, **a** holds a reference to itself. Even if it is not reachable from outside, its "refcount" will never drop to zero. Now you have a memory leak; congratulations!

It is possible to implement automatic memory management using reference counters only. Perl 5 almost does this, for instance. But memory leaks aren't good, so more often than not an elaborated garbage collector (GC) is a highly welcome addition.

A garbage collector's goal is to identify objects that are no longer reachable, and reclaim their memory.

Trace. Mark. Sweep.

A garbage collector's goal is to identify objects that are no longer reachable, and reclaim their memory. You can argue that reference counting already does this; it's true. In fact, it is one of the simplest garbage collection algorithms. Still, it's possible to implement garbage collection with no reference counting at all.

Tracing garbage collection is a good example. As the name suggests, it traces objects reachable from so called "roots" (global and stack variables). Everything else is considered garbage and deleted. The process is depicted on Figure 2.

With tracing garbage collection (let's call garbage collection GC from now on), memory isn't reclaimed immediately when an object becomes unused: instead, a dedicated routine frees it when it gets a chance to run. Quite often, when GC starts, all other operations are halted to keep the heap unmodified. This is known as a "stop-the-world pause", and is rather harmful. More sophisticated concurrent and incremental garbage collectors exist that keep pauses at minimum. For simplicity's sake, we won't touch them in this Core Tech.

When GC starts, it visits objects reachable from the roots first, and sets some flag on them. This is a "mark phase". Then it loops over *all* objects in the heap, clearing the flag and freeing those not having it

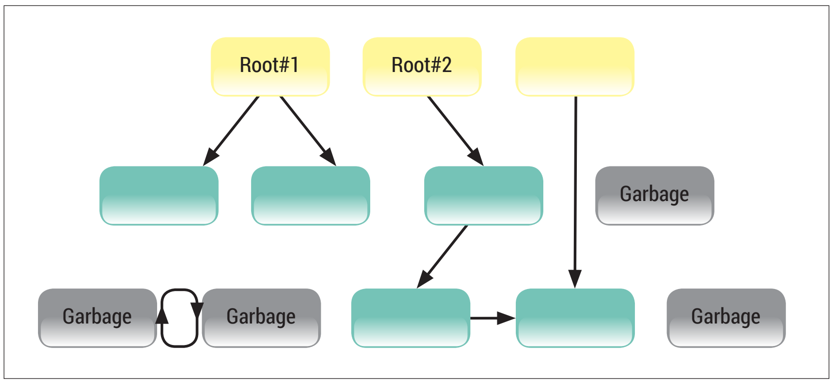


Figure 2: Tracing GC sample. Yellow boxes are roots; green ones are objects that have been marked. Grey boxes are unmarked objects, hence garbage.

Memory allocation

Garbage collection and memory reclaiming are naturally coupled to memory allocation. So how does Python gain memory for its objects?

For objects larger than 512 bytes in size, Python simply uses the **malloc()** and **free()** functions from **libc**. Smaller objects are allocated in 256kB page-aligned chunks called "arenas". On Linux, arenas are created with the **mmap(2)** syscall as anonymous mappings to avoid heap fragmentation (LV018). As objects are deallocated, their arenas may become unused. In this case, memory is unmapped and returned to the system.

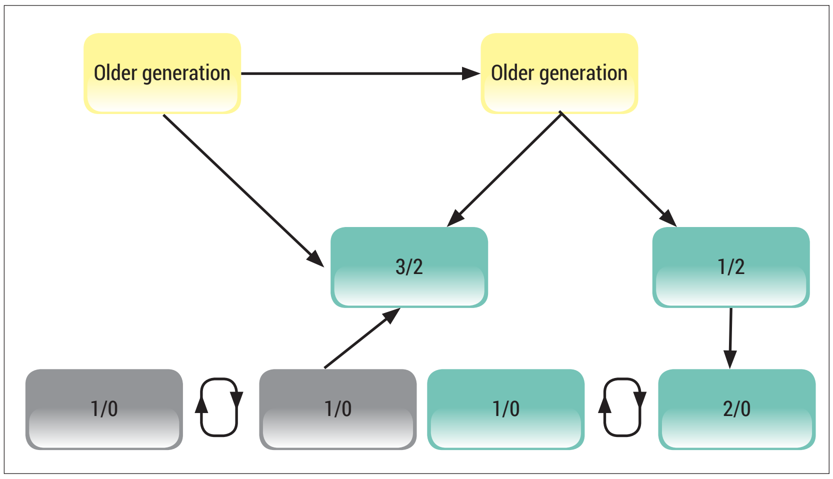


Figure 3: An example of Python's garbage collection process. **n/m** are GC refcounts before and after subtraction. Grey objects are dead, and green ones will stay alive in the next generation, shown in yellow.

set. This is a "sweep phase", and the whole algorithm is called "mark-and-sweep GC".

Some variations are possible. First, you can copy reachable ("live") objects into one contiguous memory area. This doubles memory requirements, but eliminates the sweep phase. Memory fragmentation is also reduced, but you need to spend time copying possibly large objects.

Another trick is to introduce "generations". An object that has already survived some sweeps will probably stay in memory longer than others. New objects are created in the first (youngest) generation, and each sweep promotes them to next generation until the oldest one is reached. Collection usually starts at the youngest generation, and stops there unless the system is starving for memory. This way, fewer objects need to be collected, resulting in smaller pauses.

Python objects may have finalisers or destructors, which are functions called when the object is deallocated.

Now that we've looked at the basics of how garbage collection works, we'll dissect the inner workings of Python's garbage collector.

A Python's way

As you already know, Python (or, strictly speaking, CPython) uses a garbage collector to free objects in reference cycles. Simple types like integers or strings do not produce cycles, as they don't hold references to other objects. So, GC is concerned only with containers (like tuples or lists), objects, functions and generators. Functions hold references to local and global variables, and can also grab outer scope variables via closure. This makes them less obvious, but real candidates for garbage collection.

Python already has reference counting, so its GC is naturally counter-based. Actually, each GC-tracked object has two counters: when Python decides it's cleaning time, the original refcounts are copied to GC counters. The collector then merges all generations younger than the one being collected together. Then, it follows object references within the generation and decrements GC counters for each object it visits. When it finishes, some GC counters will have non-zero values. This means these objects are reachable from outside, and should stay alive. So, GC moves them (and objects they reference) to the next generation.

All other objects are unreachable. Note however that their original reference counters were non-zero, as they were staying alive. This means that these objects form reference cycles (see Figure 3). By construction, no object that's staying alive references them now, and it's safe to destroy these cycles. The order of destruction is undetermined – in practice, the garbage collector just "clears" objects, breaking links between them. This way, their original reference counters eventually reach zero and objects are freed.

Python's garbage collector has three generations. As usual, there are no guarantees as to when the GC will run. Generally, it happens when the size of the youngest generation grows above a threshold (700, by default) or when a third-party C extension or the program itself decides. Very few programs and extensions run garbage collection themselves; most often you rely on runtime to do the right thing. Time-critical application are notable exceptions. Sometimes, they run with GC disabled (Python makes this possible) and clean memory at appropriate times. In theory, you can ditch garbage collector altogether: just be careful not to make reference cycles in your Python code, or break them manually.

To control the garbage collection process, Python provides the **gc** module. Be careful with it, as if you call its functions without thinking, you can easily screw up your program. Perhaps the most obvious thing you can do with the **gc** module is to disable garbage collection, with the **gc.disable()** function. Even when garbage collection is off, you can run it manually with **gc.collect([n])**; **n** is the generation number and is optional. **gc.collect()** returns the number of unreachable objects it found. Play with it, then re-enable garbage collection with **gc.enable()**.

Weak references

Finalisers aren't the only things that a garbage collector should take care of. Another such thing is weak references implemented in the **weakref** module.

Weak references are like normal (strong) references, except that they don't increase the reference counter for an object they point to. This means that objects having only weak references are eligible for garbage collection. When this happens, weak references don't become dangling; instead, they return `None` when dereferenced. Weak references are quite useful for caches that access objects but don't care about managing their lifetimes.

When an object having weak references is collected, Python generally does two things. First, it clears the weak references. Second, it determines if it is safe to run weak references callbacks, and executes them if possible. By definition, weak references never prevent an object from being collected, as destructors do sometimes.

There are also some tunables. **gc.set_threshold()** sets GC thresholds for each of three generations, if 700 seems too much. Changing this affects collection frequency and the amount of memory your program consumes. **gc.set_debug()** makes garbage collection verbose, which is useful when you experiment with it.

Uncollectable stuff

The previous explanation of a garbage collector's operation implied that you can free objects in reference cycles in an arbitrary order. This is often the case, but there is one notable exception.

Python objects may have finalisers or destructors, which are functions called when the object is deallocated. In the simplest case, they are implemented as **__del__** methods:

```
class C(object):
  def __del__(self):
    print('I am dying...')
```

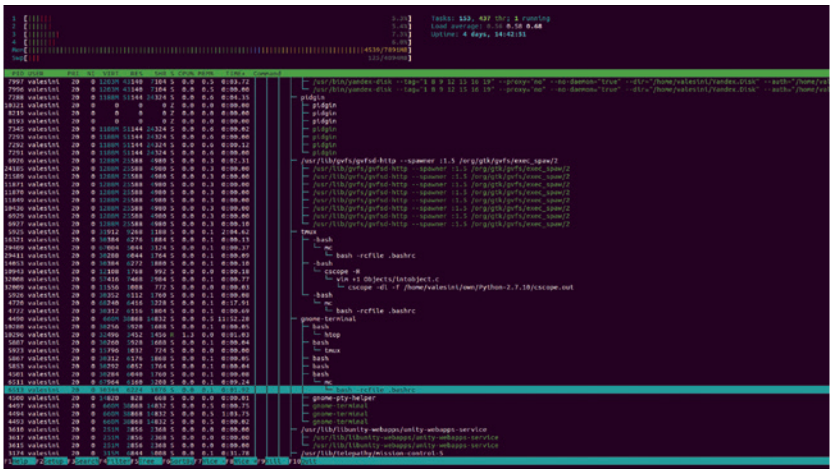
Finalisers are meant to free resources like file handles or sockets. For instance, the **pymongo.cursor.Cursor** destructor closes the cursor on the database side. However, a plain context manager (**with instance: ...**) is often a better alternative to a destructor. Some tutorials may even advise you against using finalisers anywhere in your code. So what's wrong with them?

The problem is that finalisers may contain arbitrary code, which possibly references other objects. Think of the following example of two objects in a reference cycle:

```
class C(object):
  def __del__(self):
    print('I am dying, my dear cousin %s' % self.cousin)

a, b = C(), C()
a.cousin = b
b.cousin = a
```

Suppose **a** is destroyed first. Now, the last reference to **b** has gone, and it is also freed. This calls **b.__del__()** and bang! The interpreter crashes, because **b.cousin** is a dangling pointer.

To prevent this from happening, Python (up to version 3.4) refused to collect objects with finalisers. They became "uncollectable garbage" and GC stored them in the **gc.garbage** list, so you break references manually:



With **htop**, you'll easily track all processes on your Linux box. If you remember the days of two-panel file managers, you'll feel right at home.

```
>>> del a, b
>>> gc.collect()  # Attempt to collect
>>> gc.garbage
[<__main__.C object at 0x7f785a28ce10>, <__main__.C
object at 0x7f785a29d090>]
>>> gc.garbage[0].cousin = 'Impostor'
>>> gc.garbage[:] = []  # Clear references in gc.garbage
I am dying, my dear cousin <__main__.C object at
0x7fedf7687e10>
I am dying, my dear cousin Impostor
>>> gc.garbage
[]
```

We reset the first object's `cousin` to break the cycle. Now **a** and **b** refcounts become zero and both destructors run. Note that this won't happen without some help from our side. A destructor that belongs to object in a cycle can in fact never run! You avoided the crash but got a resource leak.

You may now decide not to use **__del__**, but unfortunately it's only part of the story. **__del__** is just a tip of an iceberg, as Python really looks for a C level destructor called **tp_del**. Generators implement it, among standard Python objects. If a generator involves a **try ... catch** block, or in fact anything but a loop, and it was started, it needs finalisation. If such a generator appears in a reference cycle, it won't be collected. And as many popular Python network libraries (like Tornado or Twisted) rely on generators to make callback-based code look like sequential code, this could be a problem – but not if you're using Python 3.4 and up. Hurray for updates!

Command of the month: htop

When you want to monitor a process (including its memory usage), you'd probably use **top**. Everyone knows about **top**. So this month we nominate lesser-known yet friendlier alternative – **htop**.

htop is like **top** but with a fancier look and more natural (if you ask me) key bindings. The interface mimics *Midnight Commander*, with various features mapped to `Fx` keys and listed in the bottom bar. You can view the process list flat or as a tree, show and hide threads (including kernel ones), sort processes by whatever column you need, filter them, and send signals. Columns and meters are also adjustable, so if you want to see, say, a session ID (LV019), it's easy to do. Bars and the process tree are updated live, and if you need a quick answer on what is hogging the processor or memory, **htop** is here to help.

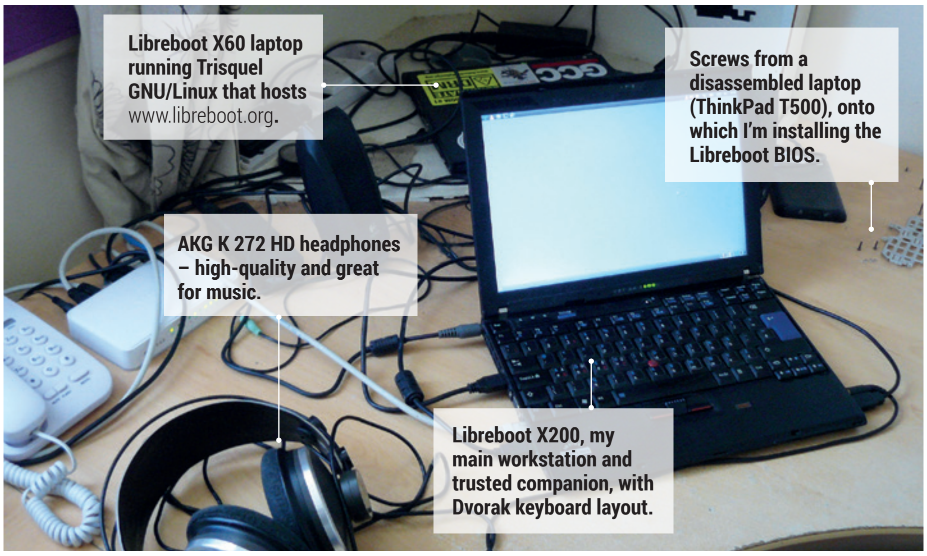
/DEV/RANDOM/

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!

It is a well known fact that the world's top supercomputers run Linux. I haven't checked the list recently myself (be my guest: **www.top500.org**) but it has been true for as long as people have made lists of the fastest computers in the world and I don't see that changing anytime soon. What wasn't quite so well know, until recently, is that probably the world's fastest distributed process also runs on Linux. Unfortunately, it is a botnet that mainly concerns itself with DDoS attacks. If you thought botnets were the domain of less secure OSes, in general, you are correct. The discovery of the malware behind XOR (**https://goo.gl/DB4bVN**) and the attacks it is responsible for puts it in a new league of effectiveness. But the more disturbing news, as security folks scramble to find wares and services to market in its wake, is how it is spread.

Basically, by brute force attacks on poorly secured SSH servers. The weakness is not per-se in the OS, but in the users (or at least the OEMs). There is, at the time of writing, no patch for 'stupid'. Many of the compromised devices probably fit into the 'appliance' category, where some embedded Linux has been shovelled on to network facing devices with little thought about robust security. Many users may not even be aware their appliance even has an active SSH server, let alone how to protect it. I'm not suggesting that patching such devices, adopting better practices and warning users is a bad idea, but it won't fix the problem. Cloud-based filtering and other DDoS mitigation may save the most prepared sites, but the XOR botnet is likely to continue to be as effective as its supercomputer cousins.


Libreboot X60 laptop running Trisquel GNU/Linux that hosts www.libreboot.org.
Screws from a disassembled laptop (ThinkPad T500), onto which I'm installing the Libreboot BIOS.
AKG K 272 HD headphones – high-quality and great for music.
Libreboot X200, my main workstation and trusted companion, with Dvorak keyboard layout.

MY LINUX SETUP FRANCIS ROWE

Lead developer of FOSS BIOS replacement Libreboot, and also runs the **www.minifree.org** shop.

Q What version of Linux are you currently using?

A Trisquel GNU/Linux, version 7.0 LTS. This is a fully free software distribution, with no binary blobs, either in the default install or in the repos. The Free Software Foundation staff use this in their offices. Richard Stallman also uses it.

Q And what desktop are you using at the moment?

A A highly tweaked Gnome "fallback" desktop, with 3D effects re-enabled. This is a very practical version of Gnome, which Trisquel includes. I've further modified the one that I use, so that it more closely resembles the Gnome 2.x desktop that I'm more comfortable with. I've also used *Xfce* and *Openbox* in the past, but I find Gnome to be highly practical, with some tweaking.

Q What was the first Linux setup you ever used?

A I first started using OpenSUSE when I was 14. Over the years, I also gravitated towards other distros: Fedora, Ubuntu, Debian, Puppy, Yopper, Arch and so on. They all have their strengths and weaknesses. Over time, I settled on Debian, which I used as my main distro for many years. Since early 2013, I have used Trisquel, one of the distributions that are fully endorsed by the Free Software Foundation, and based on Ubuntu so it still has the Debian base that I'm familiar with.

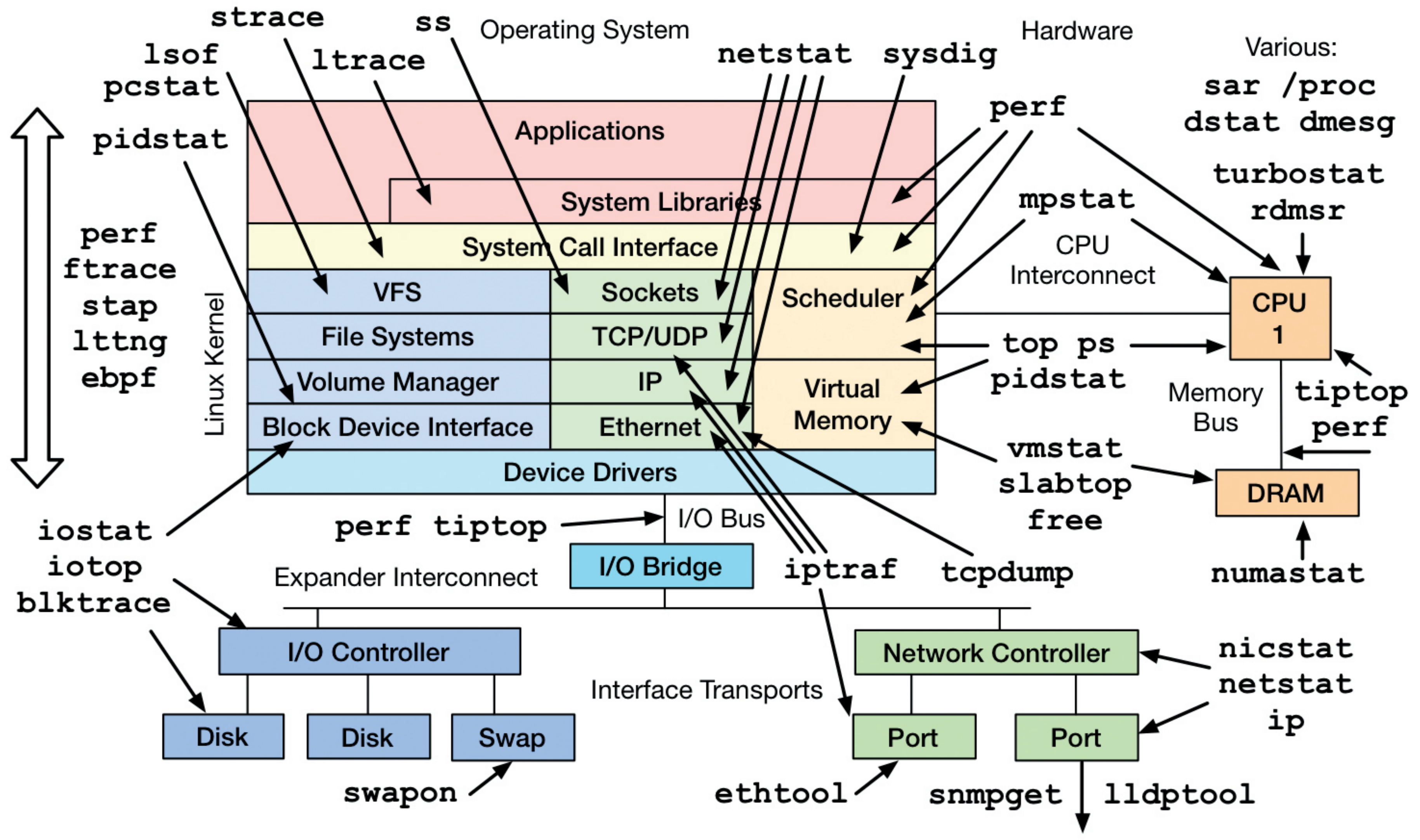
Q What Free Software/open source can't you live without?

A Libreboot! I won't use any system unless it has a fully free BIOS. I'm also the lead developer of Libreboot.

Q What do other people love but you can't get on with?

A KDE. It just sucks.

Linux Performance Observability Tools



http://www.brendangregg.com/linuxperf.html 2015

O'REILLY®

OSCON

26–28 OCTOBER, 2015
AMSTERDAM,
THE NETHERLANDS

"I'm a believer! The breadth and depth of technical know-how is incredible. This conference pays for itself."

— TRACY LEFFLER,
OSCON 2014 ATTENDEE

Save 20%
on your ticket
(use code **AFF20**)

The power of open source

OSCON in Amsterdam celebrates, defines, and demonstrates the best that open source has to offer. From small businesses to the enterprise, open source is the first choice for engineers around the world.

osconeurope.com | @oscon | #oscon