

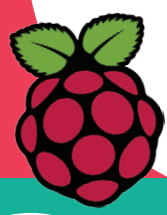


PROUDLY INDEPENDENT SINCE 2013

LINUXVOICE

RASPBERRY PI 3

Hands-on with the most powerful Pi ever



May 2016

FREE SOFTWARE | FREE SPEECH

www.linuxvoice.com

DESKTOP SHOWDOWN



Discover the perfect desktop with our ultimate guide

- OPENSTACK Build your own enterprise-level cloud platform
- WORKRAVE Protect the soft parts that you use to interface with the world
- GO Use Google's own language to create the programs of the future

32 PAGES OF TUTORIALS

ALAN POPE
MR COMMUNITY
 Inside the mind of top Ubuntu podcaster and Canonical community manager chap



WE, THE PEOPLE
THE LV MANIFESTO
 We hold this truth to be self-evident: Free Software needs a firm kick up the bum!



ZEPHYR > NETSURF > OPENSOT > SOLUS & MORE!

FREE SOFTWARE | FREE SPEECH

May 2016 £5.99 Printed in the UK

9 772054 377001

ISSN 2054-3778

054



YOUR AD
HERE



Email andrew@linuxvoice.com to advertise here

WELCOME TO LINUX!

The May 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.

Microsoft has been in the Linux news quite a bit recently. Firstly, by announcing that its SQL Server will run on Linux; and secondly by announcing its own specialised version of Debian Linux. Quite apart from what people may or may not think about Microsoft getting into Linux, this is a remarkable and monumental point in our favourite operating system's history. If you were ever looking for a sign that Linux has made it, this is it.

The question I'm now often thinking is, "What next?" Where's the next big challenge. Ben mentions in his feature on the new Raspberry Pi (p22) that more children are taking up computing at school. Open source is now the default. Even open data and open government have become important keywords. But for me it's quite clear: the challenge is now holding on to this openness in the face of big corporate investment.

Graham Morrison
Editor, Linux Voice

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, Juliet Kemp,
Vincent Mealing, Simon Phipps,
Les Pounder, Mayank Sharma,
Amit Saha, Valentine Sinitsyn

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

- 1 At the end of each financial year we'll give 50% of our profits to a selection of organisations that support free software, decided by a vote among our readers (that's you).
- 2 No later than nine months after first publication, we will relicense all of our content under the Creative Commons CC-BY-SA licence, so that old content can still be useful, and can live on even after the magazine has come off the shelves
- 3 We're a small company, so we don't have a board of directors or a bunch of shareholders in the City of London to keep happy. The only people that matter to us are the readers.

What's hot in LV#026



ANDREW GREGORY

Ben took the trip to London, where the streets are paved with gold, to get his hands on the new and super awesome Raspberry Pi 3. Since then, he's run every benchmark he can think of on it!
p22



BEN EVERARD

Go is a programming language I've read so much about, but never had the confidence to give it a go. But our 6-page tutorial takes you from complete beginner to ace image hacker!
p84



MIKE SAUNDERS

With a new version of OwnCloud out, new online bits for LibreOffice and the excellent OnlyOffice, there has never been a better time to get away from Google Docs.
p68

**SUBSCRIBE
ON PAGE 56**





Contents

More Linux! More everything really, as long as we can play with it.

Regulars

News 06

We're still waiting for Wayland, and Debian will be late – but we can while away the time playing games on *MAME*, which has now been released under Free licences.

Distrohopper 08

All hail glorious Red Star, programmed in a single night by Kim Jon-un himself after a dream in which he learned C.

Speak your brains 10

Literate dogs, our continuing shared awesomeness, and a Russian doll of a virtualisation question.

Subscribe! 12/56

Save money, get the magazine delivered to your door and get access to 26 issues of Linux Voice, in lovely DRM-free PDFs.

FOSSPicks 58

Software that soars on the waves of freedom like Lynyrd Skynyrd's *Free Bird* would if it were an actual bird.

Core Tech 94

Asynchronous disk I/O – how your machine does something while it's doing else. We can't even walk down the street and chew gum at the same time.

Geek Desktop 98

Now we know why we love Arch users so much - they're in touch with nature! Just look at this desktop to see and then read some of Evil Nick's words of wisdom.

Cover Feature



This is the year the Linux desktop becomes too pretty, to fast, too functional for the computing world to ignore. We live in a golden age!

Interview

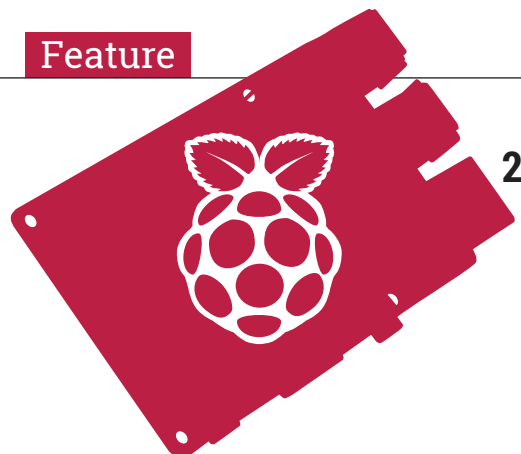


34

Alan Pope

This man fixes bugs in Ubuntu phones. He's the hero the world needs.

Feature



22

Raspberry Pi 3

Now with built-in WiFi and a 64-bit ARM processor, the Pi is tastier and more nutritious than ever.

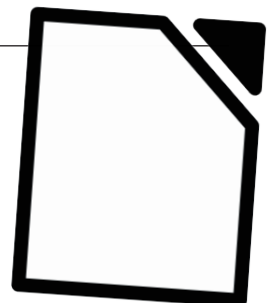
FAQ

Zephyr
The Linux Foundation jumps on another bandwagon: this week, it's the Internet of Things!

32 **Privacy distros**
The best bundles of the tools you need to keep your communications private

Group Test

SUBSCRIBE ON PAGE 56



Feature

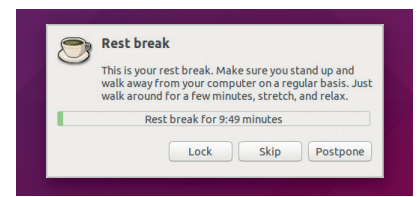
Tutorials



28

Let's fix Free Software!

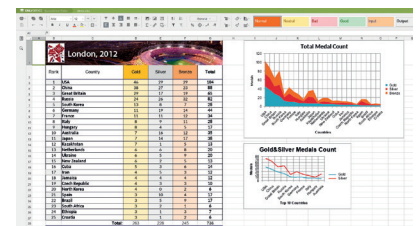
When the whinging's done, it's time to put things right. Here's how we can heal the world.



Workrave

66

Gamify your break times and save those precious carpal bones from RSI.



OnlyOffice

68

Flee Google's tentacles and set up your own cloud-based office system.

Raspberry Pi

72

Get to grips with the powerful Linux command line at the heart of Raspbian.

OpenStack

76

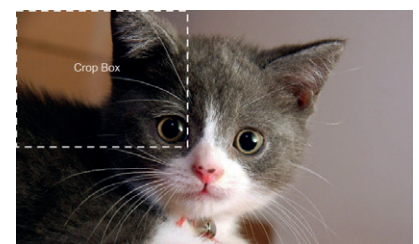
If you've 16GB of RAM lying around unused, you too can build a cloud infrastructure.

Philips Hue

80

CSS and JavaScript combine to give finesse to our lightbulb controller.

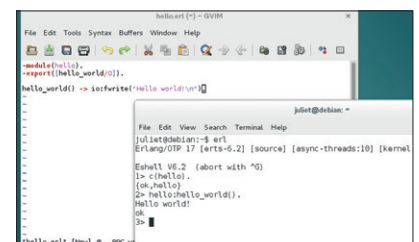
Coding



Go

84

Those clever sausages at Google have written a smashing programming language.



Older New Code

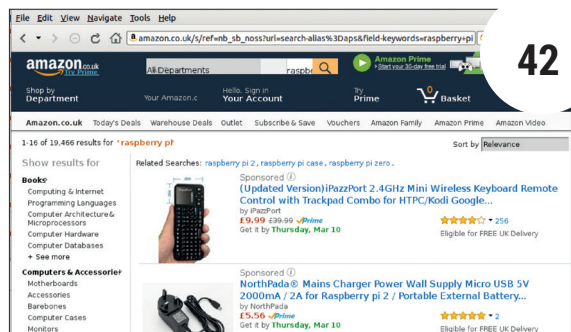
90

What's next in the world of coding? Find out with our soothsayer **Juliet Kemp**.

Reviews

NetSurf 3.4

Why borrow someone else's rendering engine when you can write your own and build a lightweight replacement for Firefox completely from scratch?



42

OpenShot 2.0

43

The power of crowdfunding brings us the latest version of this Free Software video editor.

Solus 1.1

The consumer-friendly good-looking desktop OS is back, and it's better than ever.

44

OwnCloud Server 9.0

45

This self-hostable cloud solution does everything – and now it's even taking on Facebook.



Gaming on Linux

Run, jump, shoot, plan tactics and win battles in real time against friends and foes on the internet. Also, pretend to drive big trucks across America.

46



Books

This month we're most excited about learning to code our games. *Sharpe's Opium War* is going to be the best-seller of 2017!

48

NEWS ANALYSIS

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

Opinion

Compliance?

Why is it so hard to follow the terms of a software licence? Well, it isn't...



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.

What does “open source licence compliance” mean, and does it affect you? It certainly affects VMware, which is being sued by the Software Freedom Conservancy. It also affects Versata, which is embroiled in a complex lawsuit surrounding copyrighted source code. What did they do, and should the rest of us be worried?

In the wake of those recent lawsuits I have seen more discussion of “open source licence compliance” than usual. Clearly the experiences of these companies have struck a chord, even if only with suppliers whose business depends on fear of compliance failures and mitigating that fear

Licence compliance is a major and costly issue for proprietary software, but the licence involved in that case is an End User Licence Agreement (EULA), not a source licence delivering liberties beyond mere use. In fact, mere use of open source software is never contingent on compliance; it is freely granted by all open source licences.

Of the many attributes of software freedom that could move to front-of-mind, it strikes me that the minimal licence compliance burdens for open source

software for users are actually a comparative strength. Yes, industries with strong fiduciary compliance rules will want to include their open source software in the mix with the rest of their responsibilities. But even that burden is lighter than for proprietary software. Open source licences are standardised and non-negotiable, so once you've learned about the ones in your business there's nothing more to do.

It's not a big deal...

Maybe the people making a big deal out of “open source compliance” really mean something else, then? Licencing maven Eben Moglen once shared a key insight into open source licensing. He explained that open source licences are best seen not as legal agreements but as “the constitution for a community”. While the licences we encounter with proprietary software are bilateral agreements – someone once described them as “peace treaties” – open source licences are multilateral agreements describing a shared understanding of the collaboration that's taking place.

When you use a proprietary component under bilateral licence terms, you do so knowing the intent – and prohibitions – of the copyright holder. The way you use the code is bound within the licence, and the usage and abuse of the code is clear. If you abuse the code, you can expect the next version to come with stricter licence terms or to receive preventive action from the copyright holder. If you're making an app for Apple, it may well re-express the intent once it sees how you try to circumvent it.

Using an open source component is equally bound by law. But additionally:

- The licence aims to create a set of liberties serving a community rather than a set of constraints serving a copyright holder.
- Respecting the intent of the copyright owner is a matter of community conformity rather than iterative negotiation.
- Flouting the intent of the licence is thus just as much a problem as breaching the precise definition of its terms.

When you build a business model that flouts the intent of the copyright holders, you can expect trouble. Compliance is the act of trying to flout that intent without actually facing legal consequences. The reason the VMware case in particular is so significant is that everyone involved has understood from the beginning that VMware's interpretation of the licence expressly aims to avoid needing to engage with the communities from where it derives the driver support code.

... unless you make it a big deal

This is clearly not a simple case of accidental non-compliance or even one of isolated wilful abuse, as both the Software Freedom Conservancy and the FSF have implied. It's likely that the root of the dispute is a difference in outlook. VMware sees the code as a commons to be legally ravaged, while SFC regards it as a shared community property. The way the GPL is interpreted flows from those outlooks.

Compliance – as applied to those trying to blend the GPL with proprietary code – is not primarily a matter of logic, law and licences. It's about trying to circumvent the consensus of the community behind the code with clever legal loopholes and linguistic sophistry. Is it any wonder that the community wants to fight back?

When you build a business model that flouts the intent of the copyright holders, you can expect trouble down the line

Raspberry Pi • Wayland • Steam • MAME • Mint • FFmpeg • Collabora

CATCHUP

Summarised: the biggest news stories from the last month

1

Raspberry Pi 3 is here

The Raspberry Pi has now overtaken the Amstrad PCW (remember that?) as the UK's most successful computer. Now we have a new version, the Raspberry Pi 3, which is 10 times faster than the original model thanks to its quad-core 1.2 GHz CPU. In addition, it has integrated Wi-Fi and Bluetooth – so no need to add more USB dongles! This makes the dinky device very usable as a simple home desktop machine. For Pi expert Ben Everard's assessment of the new gizmo, turn to page 22.

2

Wayland will not be default in Fedora 24

Wayland, the replacement for the ageing X.org display server, has seemed just around the corner for many years now. Fedora developers were hoping to use it as the default in their next distro release, Fedora 24, but it turns out that it's just not on par with X.org for features. It's a shame, as Wayland offers many benefits in terms of simplicity and performance, but we'll have to wait a bit longer. Wayland will still be an option in Fedora 24 for those who want the bleeding edge.

3

Steam on Linux reaches 1,900 games

A decade ago, the idea of Linux being a killer games platform was... well, laughable to be honest. But now, largely thanks to Steam, it's superb, and we have over 1,900 games to choose from. Onwards and upwards!



4

MAME released under Free Software licences

MAME, the rather awesome Multiple Arcade Machine Emulator, is one of our favourite projects, letting us relive the glory days of 1980s and 1990s arcade machines. Until now, *MAME* was released under a licence that restricted use in commercial activities – to stop arcade operators from building *MAME* cabinets, and keeping the original machines going. Now *MAME* has moved to GPL and BSD, eliminating that restriction. www.mamedev.org

5

Collabora Office 5.0 is now available

LibreOffice is great, but if you want to roll out an open source office suite in an enterprise, you'll probably want commercial support. *Collabora Office 5.0* is based on *LibreOffice 5.0* (with some extra goodies backported from 5.1) and comes with three years of long-term support. Meanwhile, Collabora is beavering away on a version of *LibreOffice* that works in the cloud, to compete with Microsoft's Office 365 offering. www.collaboraoffice.com

6

Debian 9 "Stretch" release date slips slightly

Debian often gets flak for its slow-paced release schedule, but we appreciate that it makes the distro super-solid and great for use on servers and in enterprises. The next release, Debian 9 (aka "Stretch") will now arrive a bit later than expected, due to integration of the upcoming Linux 4.10 kernel. The current plan is for the distro to go into "freeze" – so no more features added – on 5 February 2017, with the final release to follow shortly after that.

7

Linux Mint website falls foul to crackers

We often recommend Linux Mint as one of the best distributions for new Linux users, offering a polished desktop and plenty of support for extra drivers and multimedia formats (often via proprietary software, admittedly) out of the box. But in late February the Mint website was cracked and ISOs of the distro were replaced by a version containing malware. The problem was found early on, but it reminds us that Free Software is only as secure as the machines that serve it up.



8

FFmpeg 3.0 released

If you use *VLC*, *Xine*, *Blender* or Google Chrome, you're using *FFmpeg*. This provides libraries and tools for handling a vast range of multimedia formats – and it's just going from strength to strength. Version 3.0 brings about many improvements and optimisations, such as 30 new filters, VP9 hardware acceleration, a stable AAC encoder, and a Cineform HD decoder. Expect it in the next round of distro updates, or if you're feeling adventurous, grab the source code here: www.ffmpeg.org

DISTROHOPPER

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

Subgraph OS

Secure and user-friendly

The recent security issues in Linux Mint show that even the most secure of operating systems isn't always secure enough. On the other hand, an ultra-secure Linux distro that's also easy to use for the average user isn't easy to come by. That's where Subgraph comes in, offering an array of security features that are normally only accessible to more experienced users. Subgraph mitigates the resource-hungry nature of such systems, and it can be run even on lower-end machines, making it even more accessible.

The distribution is aimed at journalists, activists and others who may have good reason to require this level of security, but might not have the technical knowledge to set it up on an existing Linux distro or deal with distributions aimed at advanced users.

Subgraph comes with the Security-Enhanced Linux (SELinux) kernel module preconfigured, already saving one major headache. It also has a variety of useful features such as full disk encryption and access to an in-house encrypted mailing



Subgraph OS could prove to be hugely useful for journalists and citizens of authoritarian regimes.

system imaginatively called *Subgraph Mail*, which integrates *OpenPGP*. As expected, the distro also comes with *Tor*, which is used with any application that performs communications, through its inbuilt *Metaproxy* software, while applications allowed to make communications are limited. Another useful feature is a system called *Oz*, which isolates individual

applications from each other, so if one is exploited, others are less likely to be affected, reducing the chances of sensitive information being accessed.

A final release of Subgraph OS was supposed to be nearing completion back in 2014, but as of 2016 it has yet to move past the pre-alpha stage, though the developers indicate that things are moving along nicely.

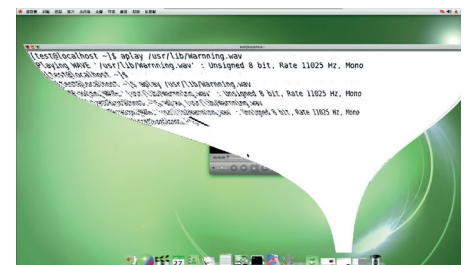
Red Star OS

Don't actually install it

North Korea's homegrown Linux distribution has been undergoing some visual tweaks, dropping the ancient default KDE 3 interface in favour of something that looks like someone was sick on OS X. The distribution is only available in Korean, there is no 64-bit version and it is far more oppressive than other state-sponsored Linux distros such as Cuba's *Nova* and Venezuela's *Canaima*, which also both feature newer software and desktop environments. The distribution consists mostly of existing software packages,

renamed with their icons changed, though some aspects have undergone more substantial changes. *Firefox* has been renamed to "Naenara" and is used to browse the country's "Kwangmyong" intranet, since internet as we know it is only available to a small fraction of the population.

Wine also comes pre-installed, presumably so that the regime could have a smoother transition when switching the country's Windows computers to Red Star. The Fedora-based system reboots itself or throws errors if the user attempts to tamper



North Koreans are apparently impressed by excessive *Compiz* effects.

with things such as the firewall, among other paranoid-fuelled additions. One thing that the regime is keen to combat is the passing around of western film and music, so Red Star also watermarks all files so they can be traced. Nice.

News from the *BSD camps

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

"Slow and steady wins the race" is the mantra behind BSD, in that updates tend to be quite unexciting, but as a result provide the stability and reliability for which the operating system is famed. DragonFly BSD saw such updates, with a 4.4.2 bugfix release after the meatier 4.4 update was released late last year. The PC-BSD developers have released version 0.8.8 of the Lumina Desktop Environment, consisting mostly of bugfixes and minor tweaks that make it more straightforward to set up, as well as out-of-the-box support for NetBSD. The Lumina Desktop aims to hit 1.0 by July of this year alongside FreeBSD 11, and will deliver a lightweight *Qt 5* desktop environment across Unix-like systems.

On the other hand, the FreeBSD team has released a report highlighting the progress made throughout 2015, and it's pretty substantial. Some achievements include assessing whether to accept GPLv3 code into the source repository, updates to MESA



The lightweight and customisable Lumina Desktop running on PC-BSD.

and X.Org, support for more ARM hardware and much more.

In the OpenBSD camp, it seems that a 14-year-old joke by developers may be

coming to an end. Website maintainers had reportedly been receiving harassing emails for years since the system uses Comic Sans Serif as the default font for HTTPD status pages and some visitors were extra-enraged when experiencing 404 errors in the font. A patch has been offered and it has become apparent that the use of the font was an attempt by some FreeBSD folks at "annoying hipsters into donating money" to the project.

The FreeBSD team has released a report highlighting the progress made throughout 2015, and it's pretty substantial

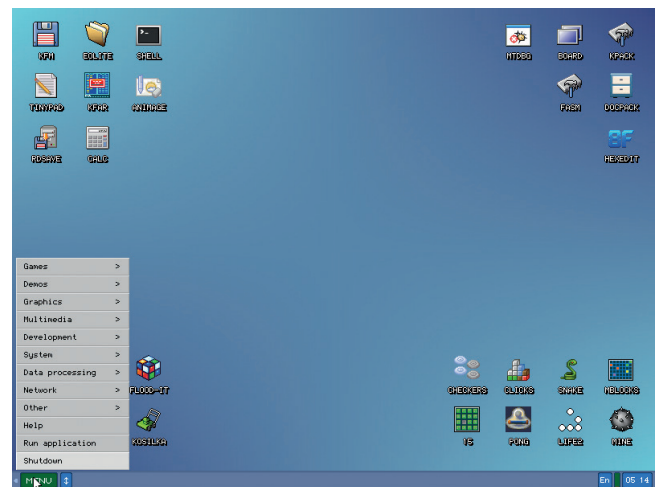
MenuetOS and KolibriOS

Digging deep into the realms of obscure operating systems, we find MenuetOS and its derivative KolibriOS. Both operating systems are ambitiously written entirely in the FASM assembly language and are incredibly tiny. (Many distributions of each operating system are small enough for their installers to fit on a 1.4MB floppy disk!) On top of this, the systems can run on as little as 8MB for RAM and 1MB of disk space in the case of KolibriOS, while providing a full graphical desktop environment. Needless to say, bloatware isn't really an issue here.

As if this weren't impressive enough already, both systems can run on incredibly old hardware (i586 processors) and still boot within a few seconds. To show off even more, KolibriOS is capable of full 1080p video and comes with more games than any other type of application, ranging from simple 2D minesweeper-like affairs to the likes of *Quake* and *Doom*. Its video drivers are based on the open source Radeon driver and are also shown off with support for eye-candy such as transparent windows.

What immediately comes to mind here is the possibility of installing this on a Raspberry Pi, though since both only support the x86 architecture, this would unfortunately not be possible. Other things which aren't possible include using either MenuetOS or KolibriOS as practical working desktops, since they lack basics like office suites, though there is a port of *NetSurf* (the go-to web browser for smaller operating systems) in the works for KolibriOS.

These ambitious projects, which are currently offered in three languages and support a multitude of filesystems, are worth keeping an eye on to track their development. Both operating systems are licensed under the GPL, though the



KolibriOS makes even the likes of Puppy Linux (which is genuinely tiny) seem huge and bloated by comparison.

64-bit version of MenuetOS is released under a proprietary licence that allows for non-commercial use only.

YOUR LETTERS

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



**STAR
LETTER**

DOGITAL RIGHTS MANAGEMENT

Every month my copy of Linux Voice arrives via by my neurotic, letterbox-guarding pet dog – always just about readable by me, but at the same time badly enough traumatised to make it unfit for any kind of redistribution.

Just one of those things or a cunning low-tech implementation of DRM via special dog attracting paper? And will I be prosecuted for circumvention if I'm caught with sellotape?

Simon French

Andrew says: Funnily enough, mine does the same. Here's a gratuitous picture of the ravening beast.



Our subscriptions team has no thumbs, so cannot take your calls. Don't phone him – turn to page 12 instead!

A WHEEL WITHIN A WHEEL

I currently use SUSE "Tumbleweed" and Mint "LMDE" at home and I have been using Linux and free software for about 20 years. While I know my way around Linux and can sort out most problems eventually, I wouldn't consider myself to be a programmer or hacker.

My work-supplied laptop is currently running Windows 7 and will shortly be upgraded to Windows 10, with all of its telemetry and privacy concerns. I have asked my work if I could run my upgraded laptop on Linux, but they have said that I must run Windows for corporate security, network, some proprietary programs and for support reasons, which also means that I can't have a Linux dual-boot option installed.

As I value my security and privacy and trust the security of Linux, as a suggestion, I wonder if you would consider a project or tutorial, whereby an existing Windows installation (eg the work supplied laptop) could be run as a virtual machine under a non-installed Linux distribution (from say a USB drive), allowing for the running of its native

Windows corporate log ons, corporate printer access and drivers and exclusive proprietary

Windows programmes when required, but use Linux for all other uses and also have the Windows privacy and security concerns addressed by Linux, eg a firewall, without having to re-boot. If this is practical, I would think that there could be other people in the situation where this approach might be of interest.

Malcom, Scone

Andrew says: Noted. We'll look into this.

BITS AND BOBS

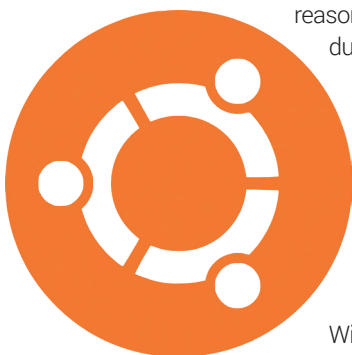
Has anyone noticed the pushing of x64 distros, particularly with those other magazines that offer a disc – hmm hope no one here started the trend, or act of sabotage! That's great and I make use of such a distro myself. No problem then. Well actually there may be. For many Windows users are using 32-bit, state-of-the-ark devices; don't tell Microsoft!

Doesn't the push for x64 distros in Linux magazines go against part of the message that we all are trying to give out: namely that it's easy to switch to Linux, which still runs brilliantly on older systems.

For many of us who are lucky enough to have newer machines then all is well. But I wonder if this switch has just come a little too early, as I see that many people are happily using older machines, running Windows XP as well! What are your thoughts on this? Does it matter?

Oscar S

Andrew says: If you're trying to evangelise, the way to do it is with 32-bit distros. A 64-bit distro is nice to have for the already converted, but the poor huddled masses on old hardware running unsupported Windows XP installations need every chance they can get of having a working system, and that means downgrading the OS to match the hardware. Even 32-bit, Linux is going to be a massive speed boost over old XP.



ELECTRONS

I just wanted to say a big thank you for offering your magazine in PDF and ePUB formats. I'm partial to PDF because you can mark it up and the markups stay with the file, unlike with the ePUB format.

I recently cancelled all of my digital subscriptions with Zinio because of numerous problems experienced over the past four years, either with their app or with account snafus. I recently tried a subscription with your main competitor via Google Play Newsstand, but I'm not able to verify my subscription, so I'm going to cancel that subscription as well. I'm glad there are some companies, such as Linux Voice, who know how to do digital subscriptions right. Keep up the good work.

Kurt Meyer

Andrew says: Thanks for the praise! We are trying our best. And in the UK, print magazines attract no VAT, whereas epubs and PDF incur a 20% tax to Her Majesty's Revenue and Customs. So we're doing the right thing (saving the trees, etc) despite being incentivised not to.



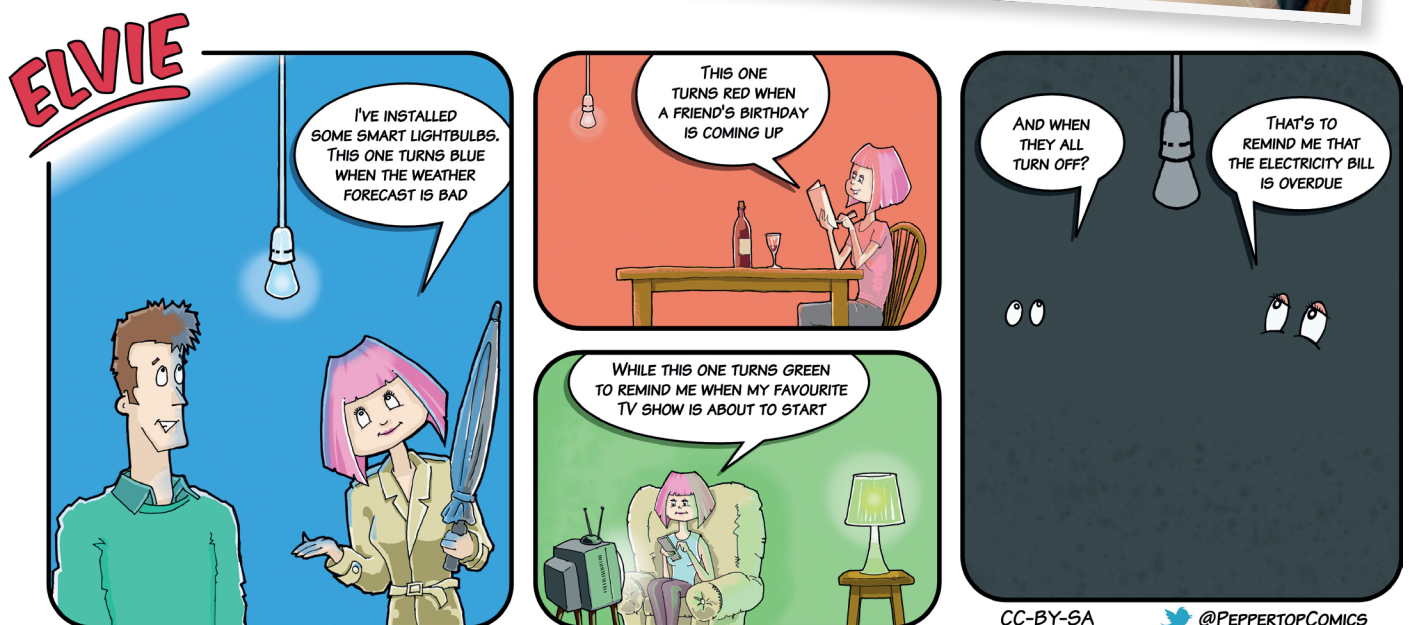
If you've subscribed, you should be able to read Linux Voice on any device you choose. That's how we do things round here!

MORE PRAISE!

I am sure you are tired of hearing how awesome you are, but I will say it again anyway. You are awesome! Keep up the good work.

And to put my money where my mouth is, I will re-subscribe to the digital version once my current one runs out. You guys need all the support you can get. Have gotten some of my colleagues and friends to subscribe also. That's it. Hope this letter encourages more people to do the same.

Gabriel Ozoani 🍷



CC-BY-SA

@PEPPERTOPCOMICS

Subscribe

shop.linuxvoice.com



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

- LV Gives 50% of its profits back to Free Software
- LV 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



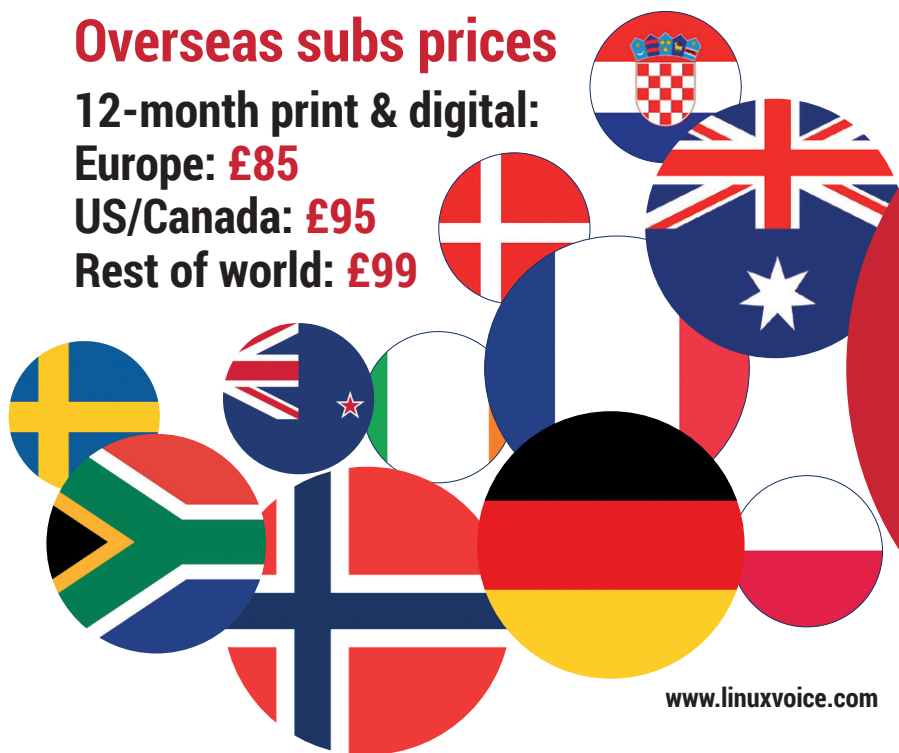
Overseas subs prices

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



2016

THE YEAR OF THE DESKTOP

We spend a month trying the most awesome desktops to work out when best to use one, two or three.

“We’ve spent the last few weeks playing with the 10 most popular Linux desktops”

Back in the olden days, there were things called ‘desktop fights’, where the merits of one desktop would be pitted against the merits of another. The results were often polarising and seldom led to any insight, and that was because there couldn’t be one ‘best desktop.’ There are just too many different ways to use your computer, and so many different kinds of users. This is what makes Linux unique – if you don’t like something, you’re encouraged to improve it or try something new.

Rather than being corralled into one way of doing something, such as with Apple’s OS X and Microsoft Windows, for example, we can all decide for ourselves depending on our own requirements – whether that’s a command line interface for a Raspberry Pi or a 3D cube running on KDE on a powerful PC. But before you can decide, you need to

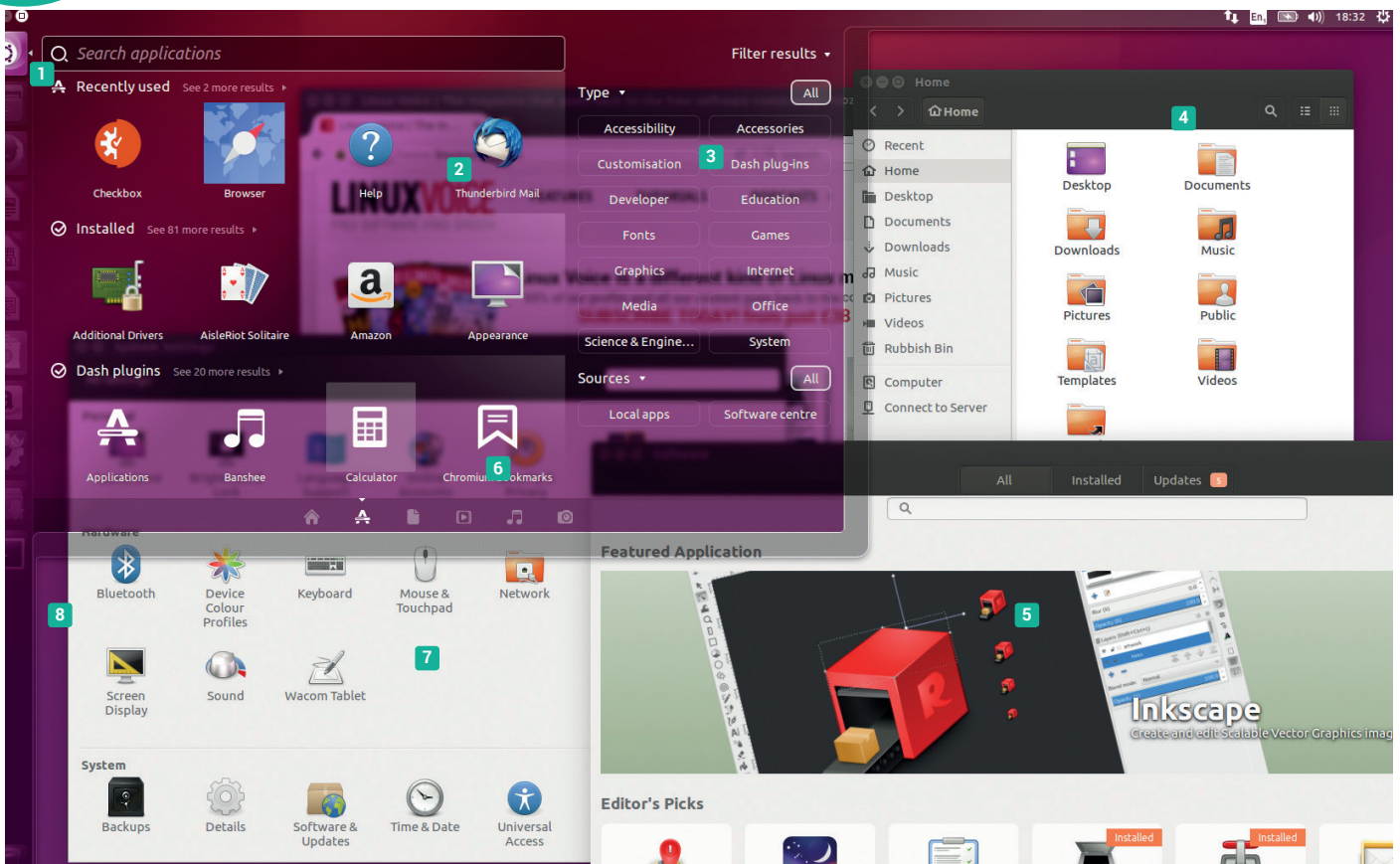
know what’s out there, and we’ve never taken a practical, visual look at what many of the most common desktops offer, until now.

We’ve spent the last few weeks playing with what we consider the 10 most popular Linux desktops, and we’ve attempted to look at each objectively, regardless of the distributions we’ve been running them on. We haven’t been trying to work out which was best, but rather we’ve looked at what kind of users may be able to get the most out of them. Most of these desktops can even be installed alongside your current favourite, especially with one of the big distributions, so you don’t need to make any significant commitment. Just give them a try and see what you think. The fact that you can do this is still amazing, all these years after the old KDE versus Gnome debate, and we can’t encourage you enough to give it a go.



UNITY 7.4

Ubuntu's desktop is moving closer to convergence with its mobile devices.



The one element that really differentiates Unity from other desktops is its all-powerful 'Dash' interface. One press of the Meta key or a click on the Ubuntu icon on the top-left and the default search scope appears, letting you quickly access your applications or files. Clicking on the small icons below lets you view applications, files, videos music and photos.

These are now called 'Aggregation Scopes', or lenses, and they're aggregating content from different sources in an attempt to make things easier to discover. The music lens has the potential to include files on Google Drive, for example, or tracks held on Amazon, and the back-ends that bring in this data are called 'scopes'. Scopes are a big part of the Ubuntu Phone operating system too, where they've become the central way of navigating through the operating system.

We're also happier now that Ubuntu 16.04 no longer enables online scopes

by default, sending your searches to Ubuntu servers and saving you from Amazon products being returned in any lenses. But it also disables useful integration with non-commercial online resources like Wikipedia, which is disappointing. If you want to turn it back on, the option can be re-enabled in the Security & Privacy panel.

Dash, Dance, Prance

Unity also does its own thing with menus and window navigation. For many applications, all menus will appear in the same place as part of the top panel, regardless of whether the application is windowed or near the top. Unless you've used OS X, which does the same thing, this can take some getting used to. But it does help Unity to feel the most integrated because most applications look and behave the same. *GTK* applications in particular, like *Firefox* and *LibreOffice*, feel just at home on Unity as they do on Gnome.

Unity

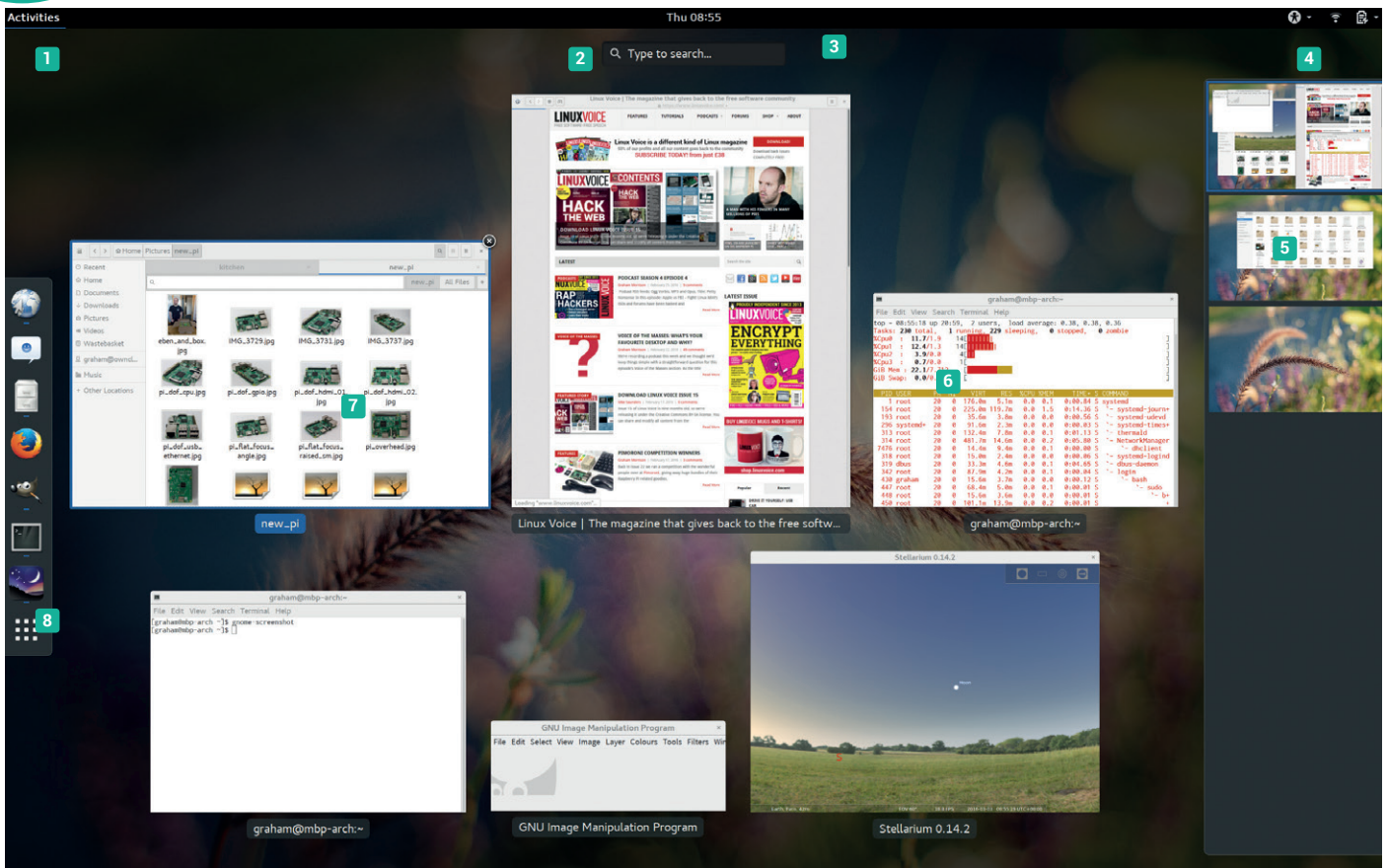
- 1 Dash** One click here and you can search everything, launch apps and explore content.
- 2 Lenses** Data is pooled from all kinds of 'aggregated scopes', a little like having Google on your desktop.
- 3 Powerful filters** Focus on specific results using a keyword search.
- 4 General settings** As with Gnome, use this quarter of the screen for networking, switching users and settings access.
- 5 App store** Not-Unity specific, but the UI and dash integration makes this a powerful feature.
- 6 Dash Plugins** Add more integration by installing plugins.
- 7 Settings** One panel is used for all the major Unity settings, including turning on online search.
- 8 Panel** Scale this and the top menu from the display config panel.





GNOME 3.18

A desktop that's easy to use, minimalist and doesn't get in the way.



Gnome 3

- 1 Activities** Moving the cursor here, or using the Meta key on your keyboard, launches GNOME's overview of everything you're doing.
- 2 Search** From the activities overview, you can search and select files and applications.
- 3 Notifications** Messages from your applications now appear here and re-appear with a click of the bell.
- 4 User control** Logout, change networks, enable the VPN and open the settings.
- 5 Workspaces** These work just like virtual desktops – drag your windows into a new space and switch quickly with shortcuts.
- 6 Apps View** See your running applications, or choose from files and apps to launch.
- 7 File manager** Nautilus is an integral part of GNOME, complete with tabs and integration with services like OwnCloud.
- 8 Panel** Choose between multiple windows and use the matrix to open the application launcher.

VHS vs Betamax. Xbox vs Playstation. iPhone and Android. Technology is littered with rivalry, and it used to be the case that GNOME and KDE were often pitted against one another. But this battle was more for the media – like Blur vs Oasis – than it was a reflection of the users and developers, and that's because both desktops have always done things differently. GNOME is for people who like to get things done, and that's because there's little in the way of distraction. By default, for instance, there are very few settings to tweak, but each and every option is considered and important, from the built-in privacy settings to the network manager.

Integration

GNOME's unique take on application launching and management, where a hotkey or screen corner is used to open a shell, shares the same roots as Ubuntu's Unity. Both desktops came from ideas shared at the same

developer's conference. GNOME doesn't have scopes nor the persistent panel on the left, and it's taken a long time to reach maturity and acceptance, but it's definitely getting there. Each release of the last 18 months has been fantastic, and we're seeing it on more and more laptops. The latest version includes Google Drive integration and a lovely new calendar; the version before that updated notifications and improved Nautilus, the file manager.

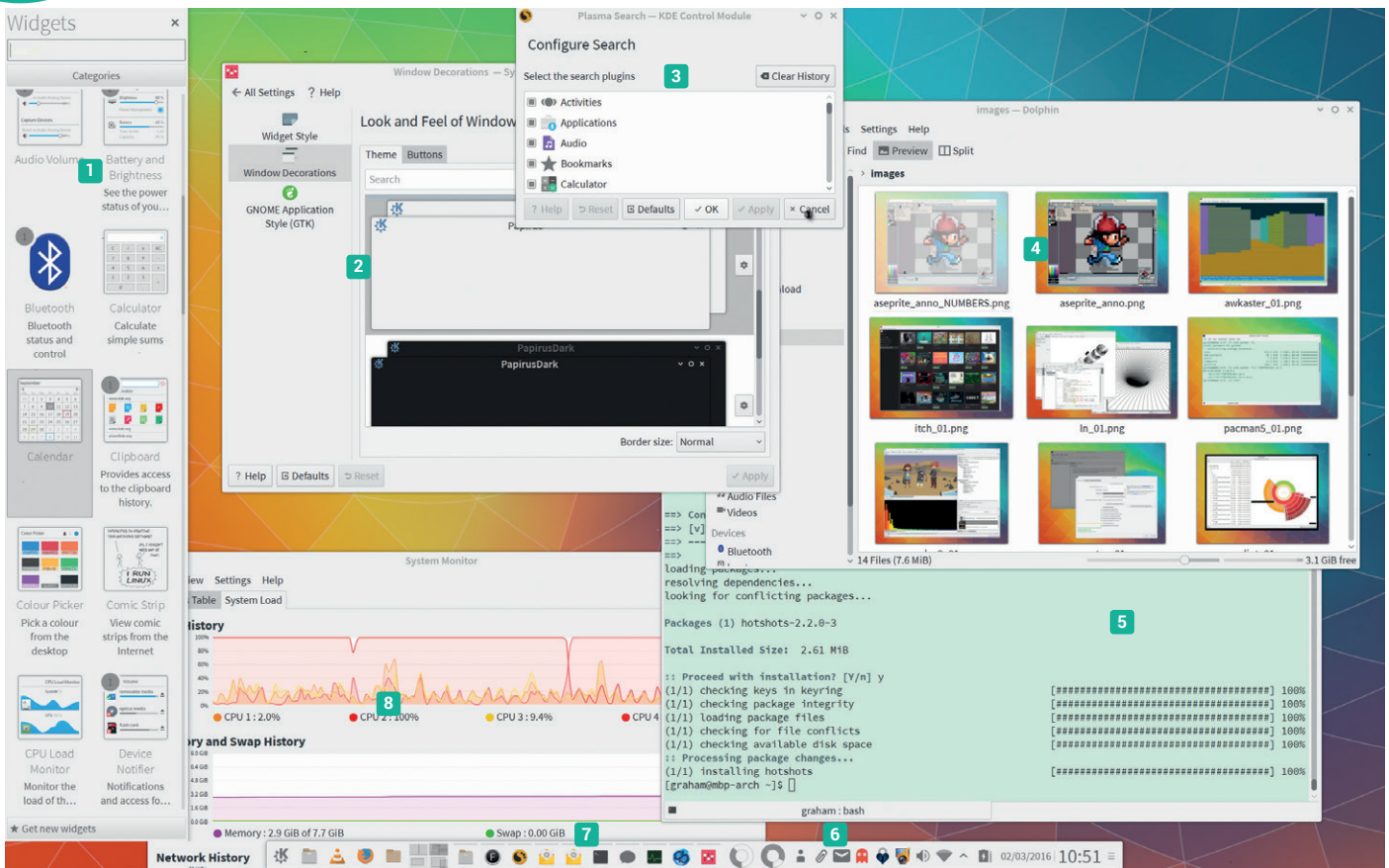
Desktop integration is another plus and one that can't quite be rivalled by other desktops. This is thanks to so many third-party projects choosing GTK, the toolkit used to develop GNOME, being used to develop their own projects, from *Gimp* to *Firefox* and *LibreOffice*. For advanced users, we also like the way the internal configuration options can be changed and updated in realtime, especially when using *gnome-tweak-tool*, giving you the best of all worlds.





KDE PLASMA 5.5

A great choice for users who want a desktop that bends to their will.



KDE is one of the two desktops that most people associate with Linux, and while both Gnome and KDE started out looking somewhat similar, KDE is the only one to have stuck with a simple launch menu and a panel at the bottom, so beloved by Windows XP users.

Many of the default choices are sensible and hide the technical aspects effectively. The Settings panel, for example, offers the KDE-flavoured control over almost everything, from changing the theme, the panel, the background and icons to adding users, configuring the VPN and integrating online accounts. Each page is well designed and easy to use, and while the icons themselves can be a little small, we feel this integration is the best of any desktop.

There's a KDE-flavoured app for almost every task, and if you're a KDE user, you'll likely prefer the way those applications are built. Over and above the ability to configure everything, this

is KDE's great strength – there's a KDE tool or application for doing almost everything, and many of these are near the best in their class.

Dolphins with lasers

The file manager, Dolphin, is the perfect example. You can quickly split views, preview multimedia files, scale the icons and sort by any number of file attributes, including image resolution and album name, if you wish. The terminal emulator is another example, and is our favourite regardless of desktop. It's tabbed with split views, has bookmarks, profiles and an infinite scrollbar. But it's the small things that make the difference, like when you're searching for a word in the buffer the console highlights new instances of that word as they appear, and these small differences are littered throughout KDE, whether it's *Digikam*, the *Caligra* office suit, *Krita* illustrator or simply re-configuring the dock.

KDE Plasma

- 1 Widgets** You can add multi-functional widgets to the background and to any number of panels.
- 2 Settings** System-wide options are all in the same place, and you can change almost anything about anything.
- 3 Plasma Search** Press Alt+F2 to search the internet and local files, look up words and perform calculations.
- 4 KDE Apps** Applications such as Dolphin that use KDE's libraries are well integrated with the system.
- 5 Konsole** The shell can be launched separately, or from apps like Dolphin and *Kate*.
- 6 Notifications** KDE supports both on-screen notification and in-panel notifications.
- 7 Panel** Add as many or as few panels as you like, floating, vertical or horizontal.
- 8 OpenGL and Wayland** Scalable drawing is possible and Wayland support is coming soon.



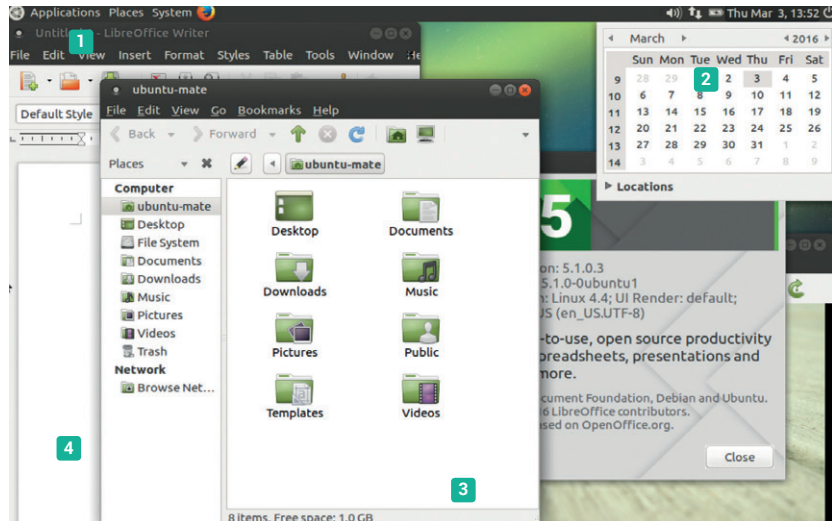


MATE 1.12.1

A retro desktop that's now going from strength to strength.

Mate (pronounced 'Mah-tay') shares a similar origin to Cinnamon, and that was Gnome 3 changing its desktop – replacing the panel and launch menu with a shell that acts much more like a launcher on a mobile phone. Mate started out as a fork of Gnome 2.x, whereas Cinnamon modified *GTK 3*, but both preserve the original desktop metaphor of the previous era. Mate, in particular, is brilliant for the many users who liked the top/bottom panel approach of Gnome 2. We tried the latest Ubuntu Mate 16.04 (a late beta at the time of writing). It's a lovely desktop that reminds you just how polished Gnome was when it wore a charcoal grey palette and chunky icons. It's quick too, reminding you that, for once, your computer is many times more powerful than it was just 10 years ago.

The desktop wasn't the only thing that forked: there are Mate versions of many of the Gnome applications, including the old file manager (*Caja*), the image viewer (*Eye of Mate*) and the terminal emulator. It may sound like Mate is more of a time capsule from 2011, but it's not. *GTK 3* applications work natively and the only real differences come from usability and aesthetics, which is something we're very happy about.



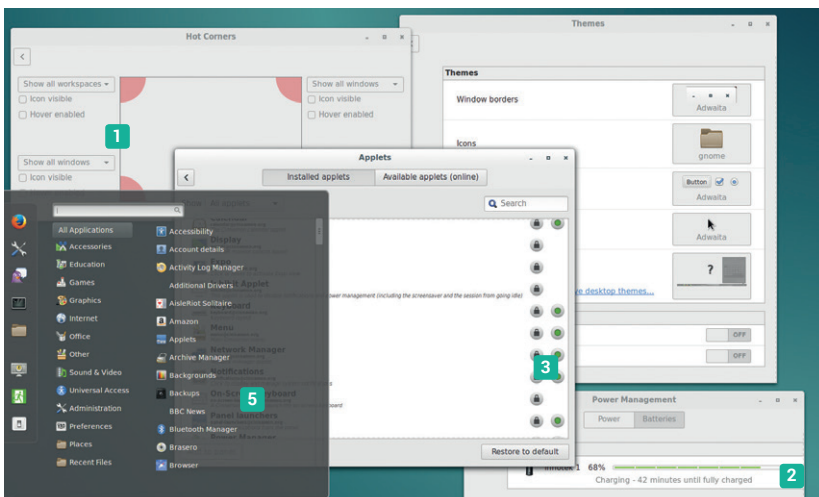
Mate

- 1 **Launch Menus** Mate looks like Gnome from 2011, and that's a good thing.
- 2 **Panels** All the same applets are available and they're all you need.
- 3 **Mate apps** There are native apps for Mate, just as there are for KDE and Gnome.
- 4 **Screen locking** Windows can be locked to parts of the screen.



CINNAMON 2.8.6

People of the world. Spice up your life!



Cinnamon

- 1 **Window management** Window snapping, hot corners and great screen rendering.
- 2 **Native apps and panels** Each version of Cinnamon brings more modern convenience.
- 4 **Applets** Not as advanced as KDE's, but more practical and less resource hungry.
- 5 **App launcher** This has a great interface, making it easy to find what you need without resorting to Unity's Dash.

Like Mate, Cinnamon was developed to maintain the old desktop metaphor. But rather than forking Gnome 2.x, it built a new desktop atop *GTK 3*, forking applications where necessary. The project was started by the Mint development team, which is where the most recent version can usually be found, but it hasn't stopped Cinnamon being ported to the vast majority of other distributions.

Thanks to using a modern API, Cinnamon's minimalism doesn't extend to its appearance or adaptability. Fonts can be scaled, for instance, and they can all be updated in real time using a slider. The theme engine is equally responsive, and while there's nothing like the amount of control you find in KDE, you can swap-out window decoration, icons, window buttons and desktops – all saveable as a theme and configurable through the settings panel. All this visual finesse is because Cinnamon uses 'Muffin', a fork of the Gnome 3 window manager, bringing together the best of the classic design with modern graphics manipulation. We also love the way panel applets can be installed and even downloaded in-line, much like you can with KDE's own downloadable GUI elements.



XFCE 4.12

The best combination for speed and visual prowess.



XFCE

1 Thunar The all-conquering file manager. **2 Eye candy** It may be austere, but there's lots of visual potential too. **3 Panel items** Add lots of new things to your panel. **4 Settings** A mature set of settings panels can change almost anything.

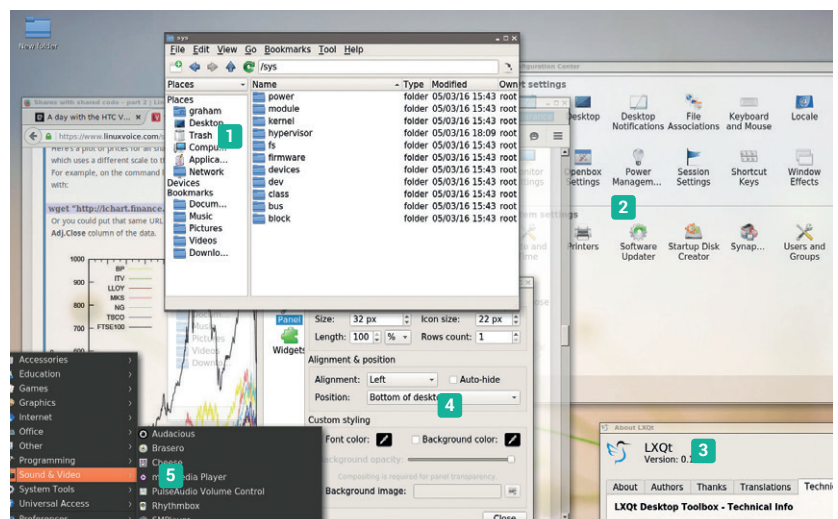


LXQT 0.10

Lightweight doesn't always have to be built around the GTK look and feel.

When it first appeared, we were hugely impressed by a desktop called Razor-qt. Built using very little but the Qt 4.3 API, it felt like a super-slim version of KDE for users who didn't need anything other than a panel and a launch menu. It didn't even include a window manager, although most of us used either KDE's KWin, or OpenBox. Which is why we're happy to see the project survives, merged with components from LXDE and reborn as LXQt.

Like Razor-qt, you have to select a window manager before starting, so this isn't the most friendly desktop, but this kind of choice doesn't bind you to your chosen environment in the way it does on Gnome or KDE, for example, even though it's perfectly acceptable to use KWin with Gnome, or Mutter with KDE. This is also the first of the really minimum desktops we've looked at. Other than providing an environment for running your applications, there's very else little to see. There's a modifiable panel and launch menu, a GUI for sudo privilege escalation, a new Wayland-compatible status notifier and a brilliantly simple file manager called PCMan. The rest of LXQt is simply an ultra-fast desktop built with a modern Qt look and feel.



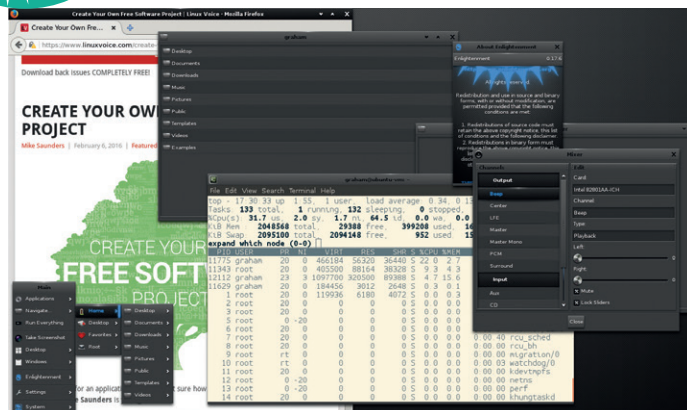
LXQT

1 File manager Fast, and works just like KDE 2. **2 Settings** Native panels for changing most options. **3 Theme and Effects** Basic but great-looking effects. **4 Powerful panel** The panel can be themed and widgets added. **5 Launch menu** Open this with Alt F1 on the keyboard.



ENLIGHTENMENT 17.6

If you miss gradients and you're looking for darkness, this may be for you.



This is a famously lightweight desktop that feels like it was designed by an Amiga demo crew. It's full of widgets and panels with shaded gradients and light flares. Everything scrolls into and out of view like part of a parallax starfield, and there are subtle animations with almost every element, from the menus to the battery widget. But this is still a powerful desktop that requires relatively few

resources, and it races along on modern hardware. You even get to select the text size before you get to the desktop, which is a brilliant feature if you're running this on a high DPI display, and there are plenty of modules and configuration options to play with. Even the window selection is unique, because you select an application by rolling your cursor over the window, which is enough to activate the process.



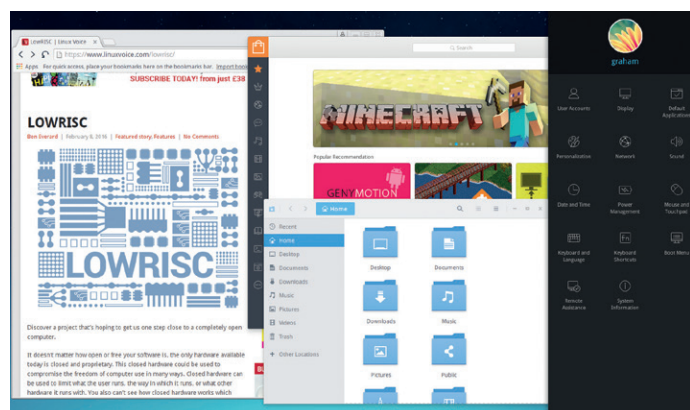
DEEPIN 15.1

Sometimes, looks are everything.



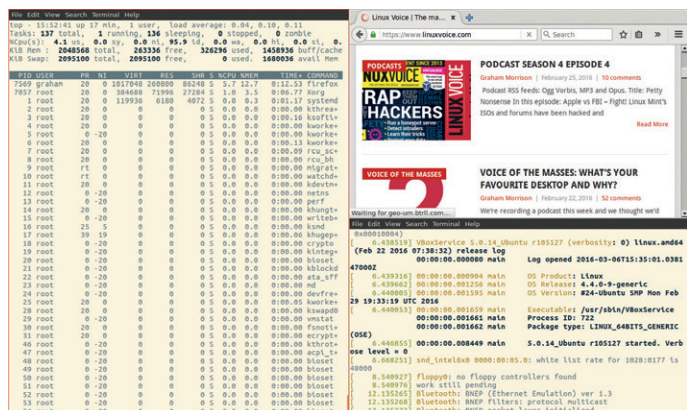
Deepin is a both a relatively new Linux distribution and an associated desktop environment. It looks wonderful and behaves quite differently to other desktops. In particular, all settings are hidden within a vertical panel that takes up the entire right side of your screen, smoothly sliding into and out of view when you need it. It's a little like the notification area in recent versions of OS X, and is

used to configure everything from two-finger scrolling on your touchpad to keyboard shortcuts. The lovely aesthetic of the user interface is carried over to its suite of native applications. There's a music player, a video player, an app store and a terminal. They all feature the same polished transitions and graphical finesse, making this one of the best-looking desktops we've ever tried.



XMONAD

Minimalism is a feature.



To finish our look at desktops, we want to include one of the several excellent tiling window managers available for Linux. These aren't strictly 'desktops' because their main feature is the lack thereof, but they perform the same function, giving you an environment to launch and manage running applications. Xmonad is written in Haskell and takes some getting used to. To start with, there's nothing to

see. Absolutely nothing. Press Alt+Shift and return and you'll get a full-screen terminal. Press that combination again to get another. Now press Alt+Space to switch between tiling modes. And so it continues. It's ultra-fast and ultra-responsive, and after you've learned a few keyboard commands, and added the xmbarr and dmenu modules, it's a brilliant way to interact with your computer.

WHICH DESKTOP IS BEST?

We recently asked our readers on LinuxVoice.com to tell us which desktops they preferred and why. We had many responses, and the answers were as varied as you'd expect, with no clear consensus on which desktop was leading, despite many commentators settling on one desktop or another. Here, we've selected a few of our favourites.

saif: "I like Unity. It seems to me so many people dumped Unity, and Ubuntu along with it, because they didn't like change. Along with this came an enormous amount of vitriol and hate for something that is what Linux is all about...more choice and a platform for innovation. Since the early days, Unity has improved and those who stuck with it helped it get better."

blahdeblah: "Unity, because Ubuntu just works out of the box nowadays, and I like boring. I've tweaked it to remove

the 300ms delay between workspace switching and a few other minor things, but it's mostly vanilla Unity."

Just working and 'boring' seemed important for users who just wanted to get on and do their thing.

Jack "Gnome 3. Just gets out the way and lets me get on with things, as well as looking good. The shortcuts and keys just work for my brain, and I really like for Super maps to Activities overview, and how I can drag and drop windows between monitors and Workspaces."

Nick: "I love myself some Cinnamon. Not just because the name summons memories of Christmassy, wintery tastes, but also because it looks sufficiently boring, so I can get some work done while not being distracted by wobbly windows. Joking aside, I switched to it, when Gnome2 was abandoned in favour of the Gnome

version that must not be named and never looked back, because it just helps me get stuff done while not standing in my way."

Electronic Penguin praised the configuration elements of XFCE: "To me it seems to... [have] the best balance between functionality and systems resources staying out of the way."

Tommi Helander and **Peter Warrington** both plumped for KDE, with Peter finding out about KDE 5 from our very own humble Linux Voice podcast. How that's public service broadcasting?

We used to wonder which desktop would come to dominate, but it has become clear that one of Linux's great strengths is that there isn't a single way of doing things, and different desktops reflect the different needs of their users. That so much choice still flourishes on Linux is something we can all be proud of, and we think, celebrate. 🐧



CHOOSE YOUR PERFECT DESKTOP



UNITY

BEST FEATURES

- 1 Brilliant integration
- 2 Consistent, clear and easy to use
- 3 Convergence turns mobiles into desktops



GNOME

BEST FEATURES

- 1 Distraction-free desktop
- 2 Easy to use
- 3 Very close to being a standard for Linux desktop environments



KDE

BEST FEATURES

- 1 Integration with many KDE/Qt applications
- 2 Configure almost everything
- 3 Looks brilliant



MATE

BEST FEATURES

- 1 Gnome at its old-school best
- 2 Fast and lightweight
- 3 Modern and well supported



CINNAMON

BEST FEATURES

- 1 The best of old and new desktop ideas
- 2 Excellent theme options
- 3 Built atop GTK 3, so modern applications fit well



XFCE

BEST FEATURES

- 1 Very stable
- 2 Thunar is a brilliant file manager
- 3 Comprehensive configuration options



LXQt

BEST FEATURES

- 1 Quick and responsive
- 2 A Qt/KDE look and feel
- 3 Transparency effects without prohibitive system overheads



E17

BEST FEATURES

- 1 Unique desktop appearance
- 2 Themes are easy to create
- 3 Stable and fast



DEEPIN

BEST FEATURES

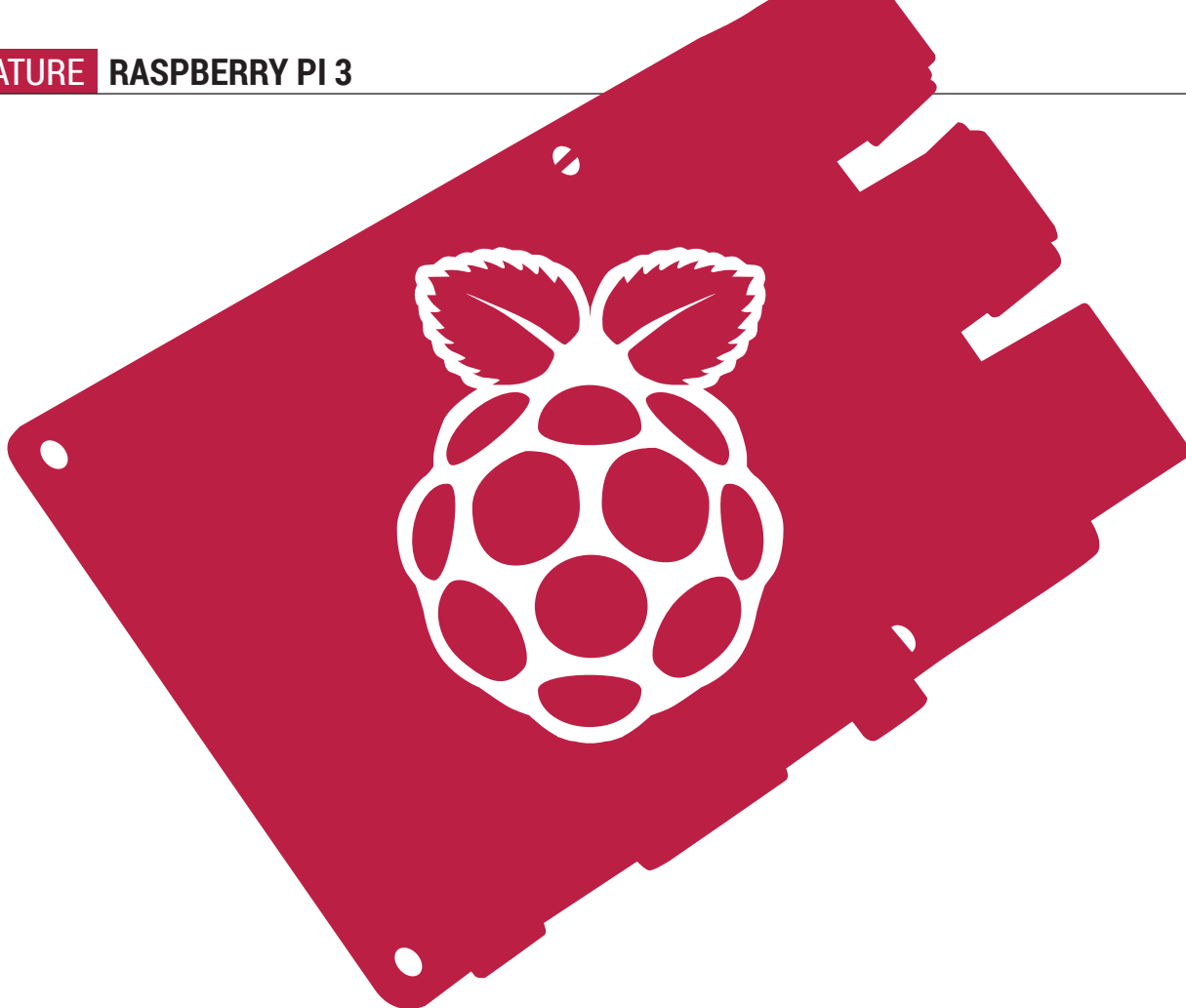
- 1 Looks fabulous
- 2 Has its own collection of custom software
- 3 Simple, unified settings make it easy to learn



XMONAD

BEST FEATURES

- 1 Tiny resource requirement
- 2 Fast and efficient
- 3 Zero distractions by graphical frippery



RASPBERRY PI

3

Four years after the first board came out, the Raspberry Pi enters the 64-bit world.

Depending on how you count leap-year anniversaries, 29 February 2016 was either the fourth or first birthday of the Raspberry Pi. Either way, four years have passed since the first went on sale and promptly crashed their online shop.

We've watched the original board evolve first into the B+, then the Pi version 2, and finally, on Monday 29 February, the Raspberry Pi Foundation released the Raspberry Pi Model B version 3. The new board promises more features and better performance for the same price (\$35 or about £27) and the same size.

Sales have continued to grow, and with 8 million units sold, the Raspberry Pi is now the best-selling British computer of all time. Unsurprisingly, these sales figures mean that there's a huge community

around the Raspberry Pi, and the computer's proving popular with tinkerers who want a small low-power computer and makers who are finding new and interesting ways to utilise the input and output capabilities in their projects.

However, the Pi is at its heart a board designed for education, and in this area there's been huge progress. Through its Picademy program, the Pi Foundation is training teachers, and we're starting to see a real impact from their educational work. The number of students taking computing at A-Level (16–18 years old) increased by almost 30% between 2014 and 2015, and while there are many reasons for this, at least some of the credit goes to the work of the Raspberry Pi Foundation inspiring students.

Through its Picademy program, the Raspberry Pi Foundation is training teachers, and we're starting to see a real impact from their educational work

THE RASPERRY PI MODEL 3

A closer look at the new board.

Place a Raspberry Pi 3 model B next to a Raspberry Pi 2 model B and it's hard to tell the difference. The most obvious change is that the power and activity LEDs have switched sides and in their place, a small, innocuous white block holds the biggest difference between the two devices – it's an aerial. The Pi 3, with its onboard Wi-Fi and Bluetooth, is the first device in the Pi family to have integrated wireless communications.

Most people with RasPis have been using USB Wi-Fi, so the onboard Wi-Fi isn't so much a new feature as a cost saving of around £10, since there's now no need to buy a separate Wi-Fi dongle. There's a second advantage in that the data through the Wi-Fi will no longer take up bandwidth on the USB bus. This leads to better performance if you're also heavily using other USB devices, for example, streaming data from the network directly onto a USB memory stick.

Bluetooth is probably less important to most people than Wi-Fi. It will enable you to use Bluetooth mice and keyboards (note: not all wireless devices are Bluetooth-enabled, so check before buying). This will also enable you to use Bluetooth audio in headphones and external speakers.

Pi for all the projects

The Internet of Things (IoT) is a tech buzz-phrase that's rarely left out of press conferences in 2016, and the launch of the Pi 3 was no exception. With Wi-Fi and Bluetooth, the new Pi is more attractive in this area than earlier models. There are two parts to the IoT: the things themselves, and hubs that are used to control the things. While the Raspberry Pi is small, cheap and low-power by computer standards, it's large, expensive and high-power by IoT standards, and this makes it unattractive for most things that could be connected to the internet. It is, however, well provisioned to make an in-home hub for a smart home setup. A Raspberry Pi could control your lighting over Wi-Fi and heating over Bluetooth, for



Eben Upton (centre) from the Raspberry Pi Foundation celebrates the launch of the Pi 3 with Claire Doyle and Richard Curtin from Element 14 (the company manufacturing the new boards).

example We don't see this being a particularly popular option initially, because current IoT devices aren't easy to connect with this sort of setup. However, previous versions of the Raspberry Pi have proved popular with small businesses developing new products, and we imagine that this could well be the same. Look out for Pi 3-powered smart home products in the near future.

The second big change is even harder to see: the new System on a Chip (SoC). There are four Cortex-A53 processing cores heart of the Pi 3 running at 1.2GHz. This compares with four Cortex-A7 cores running at 900MHz (overclockable to 1GHz) for the previous model. The biggest difference between the two chips is that the new processor is 64-bit while the

Wild speculation

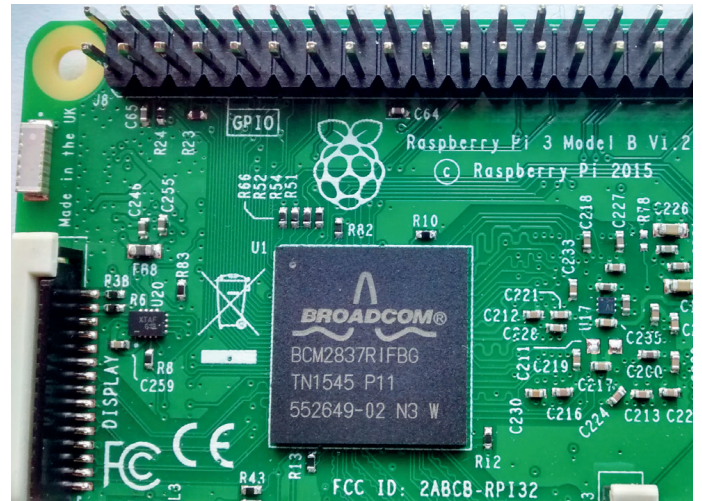
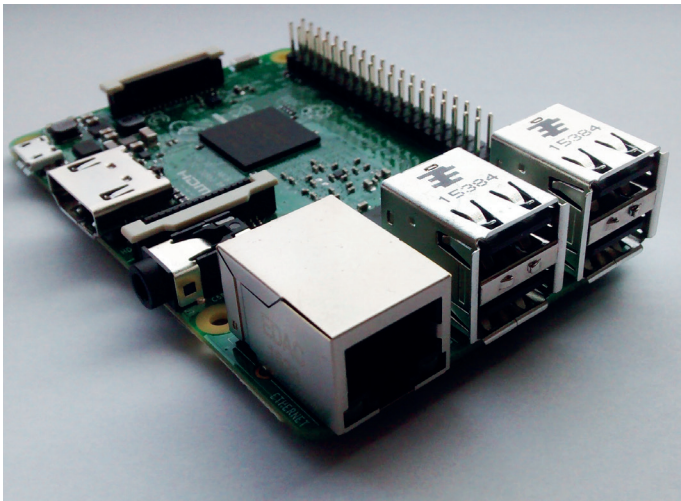
We're not privy to the internal discussions of the Raspberry Pi Foundation, but we're not going to let that stop us making some predictions about the future. The simplest is that we'll see new versions of the model A and the Compute Module featuring the new 64-bit SoC this year – this isn't really a guess, as the Foundation has already announced this on its blog. We'll add to this the speculation that the new model A will feature Wi-Fi and Bluetooth. This is a guess based on the physical position of the wireless communications on the board – they're up near the SD card, and this end of the board is almost identical between the model A

and B. The model A is most useful for robotics due to its small size and low power, so adding wireless communications to this will be a useful advantage.

The next significant upgrade to the model B will be to the memory. The Pi Foundation's been able to upgrade so much in four years because it began with quite an out-of-date processor. Even in 2012, the ARM v6 chip in the original Pi looked a little aged. The Pi 2's chip brought the device far more up to date, and the ARM v8 chip in the Pi 3 is really quite modern. It's unlikely that the Pi Foundation will be able to source a chip that is noticeably faster than the Pi 3 at the same price point for at least a

couple of years. The most obvious upgrade now is to the RAM, and we're going to guess that the next major release will be a 4GB version of the Pi 3, with a release in the first quarter of 2017.

One final piece of speculation is that we won't see a significantly improved version of the Pi Zero, at least, not in 2016. It would be too hard to hit the price point with anything faster or with more features. That, and the name suggests to us that they didn't release it with an upgrade in mind – what would they call it? A Pi 0.2? a Pi 1/2? (We wouldn't rule out the possibility of a Pi Zero + based on the same SoC but with a few new features.)



Above: The Pi version 3 retains the four USB ports and introduces power control that can supply more current to attached devices.

Above right: The small white rectangle in the top-left corner is the Wi-Fi antenna that brings new networking powers to the latest Raspberry Pi.

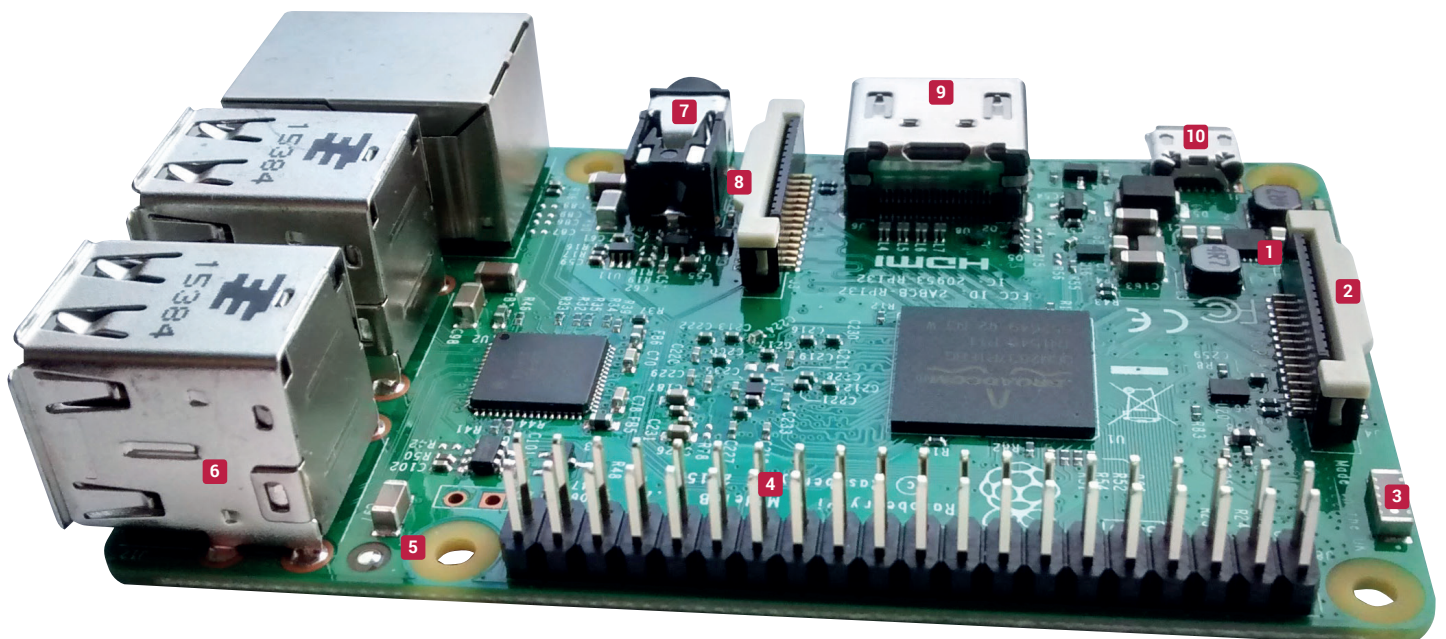
older one is 32-bit. However, at the time of writing, all the Raspberry Pi operating systems were 32-bit and running on the newer chip in backwards-compatibility mode, so don't take full advantage of the longer word length. We expect future distros to bring in full 64-bit support, which should speed up the software though we don't yet know to what extent.

The new Raspberry Pi comes with a reconfigured power supply unit that can handle up to 2.5 amps. This doesn't really affect the board itself, because it draws much less power than this, but it does enable the new Pi to supply more current to external devices, particularly USB devices. Since the release of the Pi 1

B+, the power issues for USB devices have reduced significantly, and we've found that we rarely have problems with the usual setup (mouse, keyboard and Wi-Fi). However, if you're using more than this, the extra half an amp may mean you can get by without using a powered USB hub.

Perhaps the most minor upgrade on the new Pi is to the SD card slot. On the new board, cards are pushed in and pulled out rather than pushed in, then pushed again to release them. It's a minor change, but one that will please anyone who's ever accidentally pushed an SD card on a running Pi and had it pop out and crash the Pi.

Features of the Raspberry Pi Model B Version 3

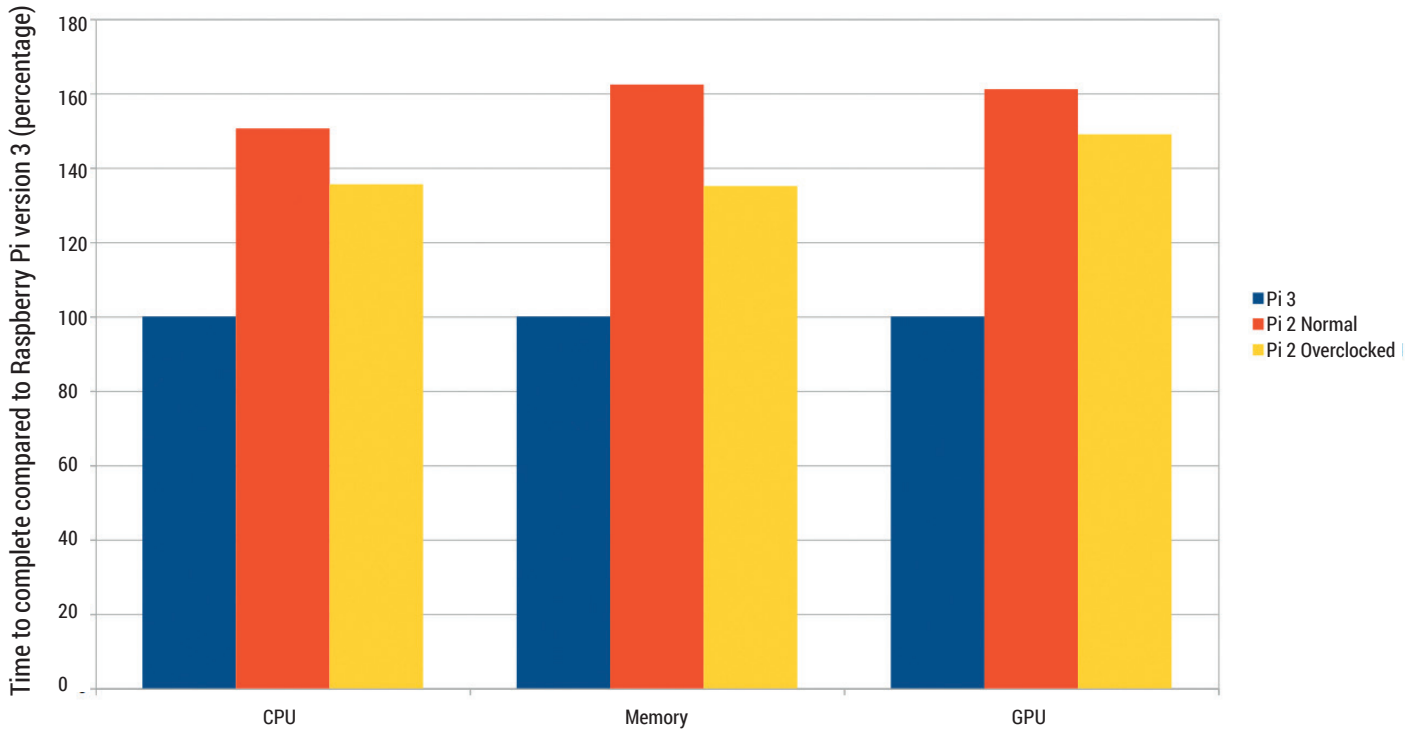


1 Power and activity LEDs have moved to make way for Wi-Fi **2** Connector for official 7-inch touch display **3** Aerial for wireless networking **4** 40-pin header with 26 addressable GPIO pins including UART, I2C and SPI functions **5** Mounting holes for Pi HAT extension boards **6** Four USB ports can supply more current than previous Raspberry Pi models **7** Four-pole connector for audio and composite video output **8** Connector for official camera module (also available without IR filter for low-light photography) **9** HDMI output **10** Micro USB power input can now handle up to 2.5A.

BENCHMARKS

We find out just how fast the new Pi runs.

Raspberry Pi benchmark performance averaged by area



The Raspberry Pi 2 took 30–60% longer to perform the benchmarks than the new Raspberry Pi

At present, the Pi 3 doesn't officially offer any overclocking, while the Pi 2 can be overclocked from 900MHz to 1GHz. We tested the performance of the Pi 3 against both of the configurations of the Pi 2.

Our first set of tests compared the performance of the key systems on the board: CPU, memory and graphics. Each of these showed a significant speed up over the previous generation. The Raspberry Pi Foundation claims that the new board is 50–60% faster than the Pi 2, and our benchmarks support this for the Pi 2 at default clock speed. Compared to an overclocked Pi 2, the speedup is about 30–40%. Since both devices are quad core, this speed increase is reflected in both single and multi-core performance.

All models of the Pi have all their networking options on the same bus as the USB ports. This means that if you were heavily using the network and USB devices at the same time, you could run into performance issues as this bus reached capacity. On the new Pi, the onboard Wi-Fi is connected to a different bus, so shouldn't cause the same level of problems. In order to test this, we streamed data over a wireless network directly onto a USB memory stick. On the Pi 2, we used a USB Wi-Fi dongle. This test simulated the performance characteristic of the Pi used as a network attached storage device (NAS) device. The Pi 3 transferred a 400MB file in 53% of the

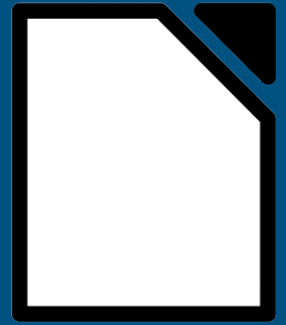
time the Pi 2 took. The wired Ethernet port on the Pi 3 still goes through the same bus as the USB, so to get this level of performance, you have to use the onboard Wi-Fi, but it makes the Pi much more attractive as a NAS controller. The new networking controller isn't going to lift the performance of the Raspberry Pi to the level that you'd expect from some of the other ARM boards that include gigabit networking or SATA hard drive controllers such as the Odroid C2 or the x86 MinnowBoard, but it does bring much better performance and enables you to keep benefiting from the Raspberry Pi software and community.

Better, better...

The new Raspberry Pi is faster than the previous model, but it's not the same step up as the transition from the Pi 1 to the Pi 2, when we saw a six-fold increase in performance. The 30–40% speedup over an overclocked Pi 2 is noticeable, but it's not enough to really change the usage of the device.

For us, the onboard Wi-Fi is the biggest feature of the new device. Unless you're specifically having performance problems, the Pi 3 doesn't offer enough new features to recommend abandoning a Pi 2 for it. However, if you're in the market for a new single-board computer, the Raspberry Pi 3 Model B should be the first device you consider. 📺

SECRETS OF LIBREOFFICE




LibreOffice Writer is capable of way more than simple text documents.

Word processors are one of the most under-utilised applications – not because they're rarely used, but because they're rarely used to their full potential. Many people just use them as text editors with better font control, but they're far more than this. A good word processor includes the features of a desktop publisher, spreadsheet, drawing tool, text formatter and Turing-complete language all into one neat package

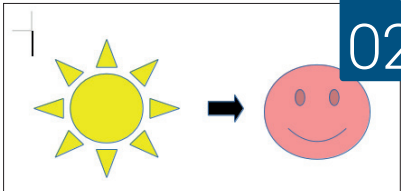
that should be simple enough for someone to pick up and use without any training. In truth, a word processor is the most complex bit of software that most computer users use.

In order to stay useable to beginners, many of the more advanced features are hidden away. Linux Voice is here to shine a spotlight and help you uncover the full power of one particular advanced word processor – *LibreOffice Writer*.

 **Text boxes help control the layout of text in wordprocessor documents**

They allow you to place text at a particular place

01



02

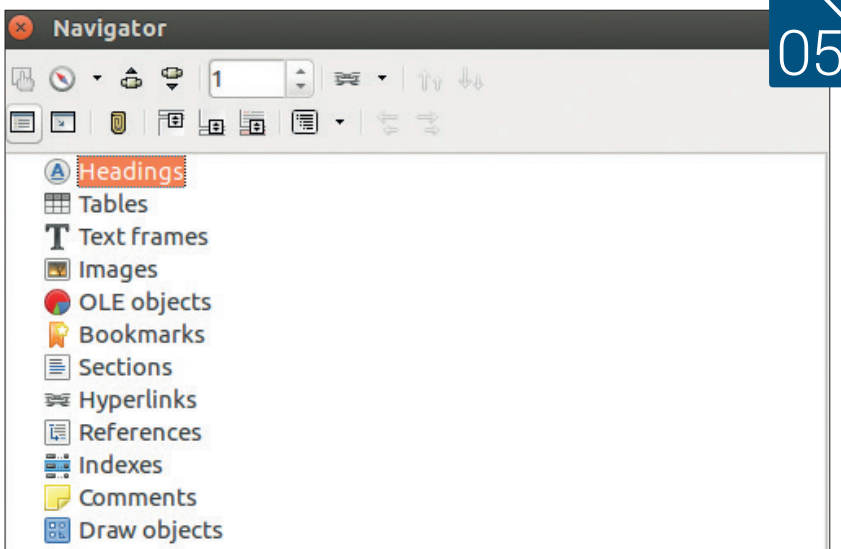
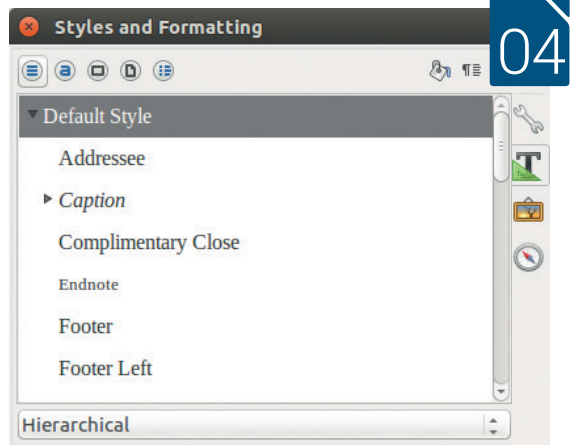
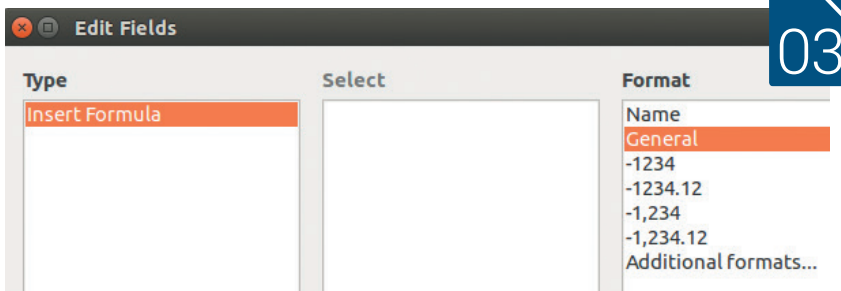
01 Layout Normal documents are laid out from top to bottom and from left to right. Adding text at the top will push all the content down, and it'll flow onto other pages if need be. This is exactly what you want for most text documents, but occasionally, you need a little more control over the layout. In *Writer*, you can add text boxes to specify exactly where you want the text to go regardless of how the rest of the document flows.

02 Drawing If you need to add an illustration to your document, there's no need to switch to a separate program: just enable the drawing toolbar and sketch from within *Writer*. The options are a little limited, but more than enough for a quick flow chart or simple diagram. By keeping all the material in a single file (rather than including a picture created elsewhere), you only have to keep track of changes in one place.

03 Formulas Need to perform a calculation in your document? You could open up a separate program, work it out, and put the answer in, but that way, you can't see how you calculated the result when you go back. Instead, you can add the Formula Bar (View > Toolbars > Formula) and embed the actual calculation in the document. Only the answer will appear in the text, but the maths will be saved so you can go back and edit it in the future.

04 Styles Proper management of styles will make your documents look a lot better. This doesn't just mean making stuff bold or italic, but creating named styles that you then apply to your text as you create your document. These named styles enable you to update particular styles across your whole document in one go, and by saving styles in

If you need to add an illustration to your document, there's no need to switch to a separate program: just sketch from within *Writer*



templates, you can ensure consistency across a range of documents.

05 Navigator

The Navigator is *Writer's* most powerful document management tool. It enables you to manage the various parts of your document from a single, dockable tool set. Using the Navigator, you can see all the aspects of your document and also move them around. For example, if you decide to switch two sections of your document, you can use the Navigator's Chapter Up function to move a heading and block of text up (or down) through the document rather than having to copy and paste it manually.

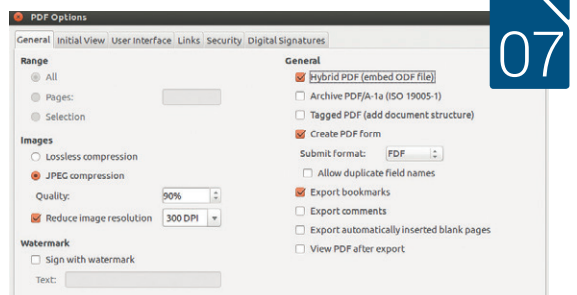
06 Emoji

Until the 3rd Century BCE, text had no punctuation. Aristophanes, a librarian at Alexandria, decided text was too hard to read without and spaces or marks between any letters, so he introduced dots between intermediate, subordinate and full points. Little did he know that he was starting a process that would lead to symbols gaining ever more prominence in text and a gradual return

to hieroglyphs. Linux Voice is reliably informed that the youth of today now mostly communicate in small icons known as Emoji. *LibreOffice*, ever keen to move with the times (even when that movement is towards anarchy), makes it easy to insert these abominations into documents by placing a word between two colons. For example, **:keyboard:** will insert a picture of a keyboard. Why that's better than just writing the word keyboard we don't know, but who are we to argue with the forces of progress?

07 Hybrid PDFs

There are two ways you can share documents: PDFs and editable formats. The former method is great because it guarantees the layout (which is especially important if the document needs to be printed); however, it's difficult to edit. The latter is useful if the document needs to be edited, however, it can render differently on different setups. *LibreOffice* allows a compromise between these two extremes – hybrid PDFs – which are normal PDF files but with the entire document embedded in an editable format that can be changed.



08 Addons

Can't find the feature you're looking for? No problem. *Writer* has a plugin system that enables people to extend the core with additional functionality. You can create your own, but it's far easier to grab ones that already exist. Take a look at <http://extensions.libreoffice.org/extension-center> for a list of available options. That website also includes a range of dictionaries in case your *LibreOffice* install didn't come with the languages you're creating documents in. 

THE LINUX VOICE MANIFESTO

WHAT'S RIGHT, WHAT'S WRONG AND HOW WE CAN CHANGE FREE SOFTWARE FOR THE BETTER

We often talk about the Free Software movement (and indeed the open source and Linux communities) as if everyone involved were part of one big, cohesive group, where everyone is marching in the same direction towards a land where software is free to share and modify. And to some extent that's true: we all have broadly similar goals. But at the same time there is so much disparity – especially when you consider all the different GNU/Linux distributions, projects, licences and personalities involved.

Now, there's definitely strength in diversity. If we just had one distro, one desktop environment and one Free Software licence, we would be at risk of our community

Sometimes it helps to state your mind instead of dressing everything up in cotton wool, but at what point does it go too far?

falling apart (eg if a lead developer were incapacitated or the single licence on which we depend was found to be unenforceable). So it's good to have variety. But do we have too much sometimes? Does the vast proliferation of desktops and distros help or hinder the GNU/Linux project when it comes to spreading the word?

Prickly characters

And then we have to think about personalities. Hackers and software engineers tend to be rather blunt and direct in their communications, sharply criticising the work of others if they feel that work is holding back a project (see the many expletive-laden rants from Linus Torvalds as an example). Sometimes it helps to state your mind and get things fixed instead of dressing up everything you say in cotton wool, but at what point does it go too far?

And why do some opinions turn into rabid hatred – like the obscenities and threats hurled at Lennart Poettering for his work on Systemd?

Many Free Software projects have introduced Codes of Conduct to mitigate this problem; some argue that they will simply turn mailing lists into “safe spaces” where nobody can even criticise another's code, and the effect will be detrimental in the long run. The goals are noble, and any attempts to make communities more welcoming are to be applauded, but shouldn't the quality of software be the most important factor in the end?

So this issue we thought we'd look at the state of Linux, Free Software and open source today, and put our take on it: what's right, what's wrong, and where we should be heading in the future. So – let's put the world to rights!

THE GOOD

Many of us began using GNU/Linux in the late 1990s, as the internet was starting to work its way out of universities and into homes. Back then, Linux on the desktop was rather rough, and we didn't have anything in the way of attractive and integrated desktop environments. No, if you somehow magically managed to get XFree86 working, you ended up with a rather ugly FVWM desktop (or its more Windows 95-esque AnotherLevel spin-off).

Rough edges aside, the GNU tools and Linux kernel were impressive achievements back then. But the big question was: is any of this sustainable? Can businesses be built around Linux and Free Software? Or is it all silly hippy commie "anti-American" nonsense that will die out when programmers realise they have to get bread on the table? (Sadly, that was the argument from a lot of the proprietary software world at the time.)

A few companies such as the German SUSE and American Cygnus Solutions had demonstrated that they could make money by selling and supporting Linux, but it was Red Hat that really took the charge and battled to get Linux into enterprises. Red Hat was generally a good community player and today is making megabucks with Linux.

But it's not just about companies. In some prominent cases, Free Software projects have been largely abandoned by their commercial sponsors but taken up by communities and not-for-profit organisations. Take *OpenOffice.org*, which ended up in the hands of Oracle – and then passed on to the Apache Foundation. Today, *Apache OpenOffice* is effectively dormant. Fortunately, just as Oracle was handing off *OpenOffice.org*, a new community-led organisation called The Document Foundation was created to work on a fork of the codebase, *LibreOffice*. Today the Foundation has enough financial support from donations that it can pay for infrastructure, developers, documentation staff and more. It shows that even without the backing of a single big company, a major open source project can still flourish.

Power to the people

At Linux Voice, that's what we'd love to see more of: communities and non-profit organisations rallying behind big FOSS projects. It's great to have Red Hat, Canonical, SUSE and co. paying developers to hack on Linux and open source apps, but companies come and go. Also, goals and objectives can change: *Firefox* started out as a lean, fast and power user-friendly



(through extensions) browser that achieved rip-roaring success. Today its popularity is plummeting, controversial add-ons such as Pocket are being chucked in to pay the bills, the interface is constantly being rejiggled and features are being removed (or dumped into **about:config**).

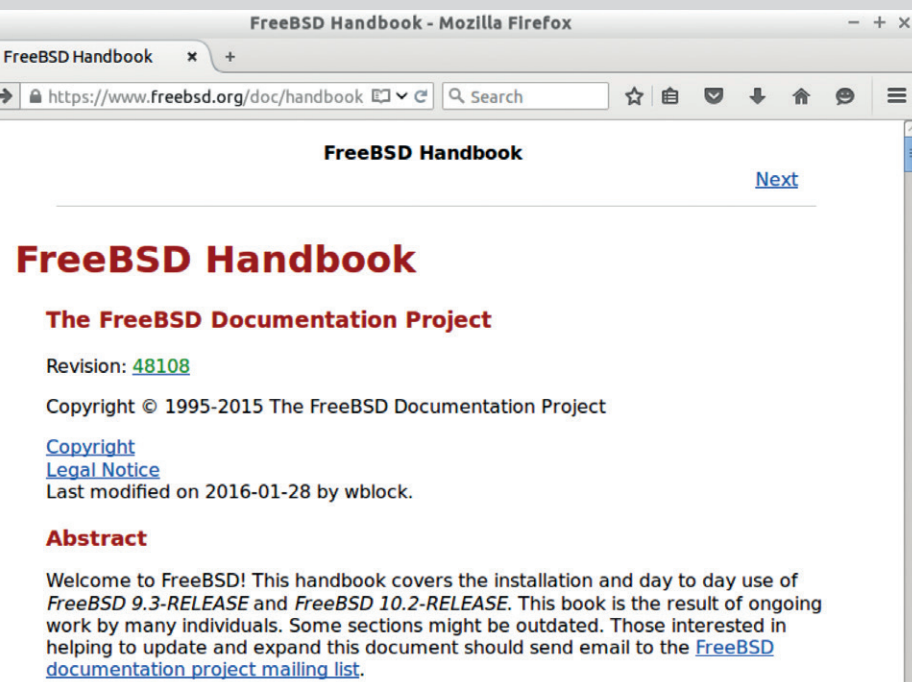
Maybe it's time for a non-profit organisation to step up and work on the code, like The Document Foundation did with *LibreOffice*. It's too early to say if such an organisation could make enough money from donations and merchandise sales to fund developers, documentation, servers and all the other bits and bobs that a project needs, but it could be worth a try.

Technology-wise, we have desktop and productivity software well covered with *Firefox*, *Thunderbird*, *LibreOffice*, *Inkscape*, *Scribus*, *Gimp*, *Audacity* and other tools. Some may still have rough edges, but it's possible to do almost every daily task on Linux. And of course, on the server it's leading the pack by a long way, so there's nothing to gripe about there.

Some people dislike Systemd – fair enough. But the sheer amount of abuse that its lead developer, Lennart Poettering, receives is worrying.

Maybe it's time for a non-profit organisation to step up and work on the Firefox code, as The Document Foundation did with LibreOffice

THE BAD



FreeBSD's Handbook is an excellent, well organised and single-stop-shop for documentation – something many Linux distros would be wise to copy.

One of the greatest things about community-developed open source projects is that people can choose the bits and pieces they want to work on. If you work for a proprietary software company as a 9–5 job, you may have some awesome ideas for the application you're working on, but your company has forced you to write boring test cases all day. *C'est la vie*. If you're donating some of your free time to a FOSS project, it's more likely that you'll work on the components that interest you the most.

There is a downside to this, though: some of the more laborious aspects of FOSS development are often neglected. Take documentation as an example. Who wants to write that, when you could be hacking new features, optimising code to run faster, or working on graphics or sound for a game? So documentation in the Free Software community is something of a mixed bag. A few projects have superb docs, but in many cases they're lacking or simply non-existent. Because Linux is developed in "Wild West"

fashion, with lots of groups (kernel, *Glibc*, *GCC*, *X.Org*) doing their own thing and distro vendors shoehorning the results together, it's hard to put together a complete and integrated set of documentation that covers every aspect of the operating system.

Who wants to write documentation when you could be hacking new features, optimising code or working on graphics or sound for a game?

Compare this to FreeBSD, for instance, which has a superb Handbook (www.freebsd.org/doc/handbook) providing a reference to all aspects of the OS in a single location.

Aggravating this problem is the variety of Linux distros out there. We have nothing against specialised custom distros that focus on a specific user, such as penetration testing distros, distros for visually impaired users and so forth. But spend half an hour browsing www.distrowatch.com and what do you see? 95% are distros that honestly don't need to exist, and could be recreated with an *apt-get* or *Yum* script.

If a distro is basically Fedora or Ubuntu with a slightly different package set and a change of wallpaper, does it really need to exist? No, we think. At least, not in the open. Tying with distros is great fun and if you make a spin of Fedora that you like – great, have fun. But please don't make a big song and dance about it, upload it to a site, add it to DistroWatch and other places. It just adds entropy to the ecosystem and makes things confusing for new users when they're trying to choose a distro. We need a small range of top-quality, distinct distros, not thousands of largely identical ones that only exist to boost the egos of a few developers.

I want my new FooApp 2.0!

Then we have the problem of software distribution. There's something fundamentally wrong when you install an OS such as Ubuntu, and have to wait six months before you get new releases of software. (Of course, you can mess around with PPAs, backport repositories and other methods, but it gets messy.) We need a simple, streamlined and effective way for third-party app developers to push a button and boom – their app can now be installed on any distro with a single click. The now defunct Autopackage project was making good progress in this direction, but few apps ended up supporting it. Since then, rolling-release distros such as Arch have exploded in popularity, but they have potential sticking points in large-scale enterprise deployments where long-term support and absolute stability is vital.

Finally, another issue that needs tackling is developer mentoring – and making the community more welcoming in general. Now, this varies from project to project; some put more effort into helping new developers get involved in the code and submit their first patches, whereas other communities are more elitist and tend to look down on newcomers.

THE ROAD AHEAD

Ultimately, we – as in Linux Voice, or indeed any other supporters and advocates of Free Software – can't force anybody to do anything. And that's a good thing. As mentioned, most FOSS developers work on their projects because they're passionate about it, so nobody can simply tell them to shift their attention to something that doesn't interest them in the slightest.

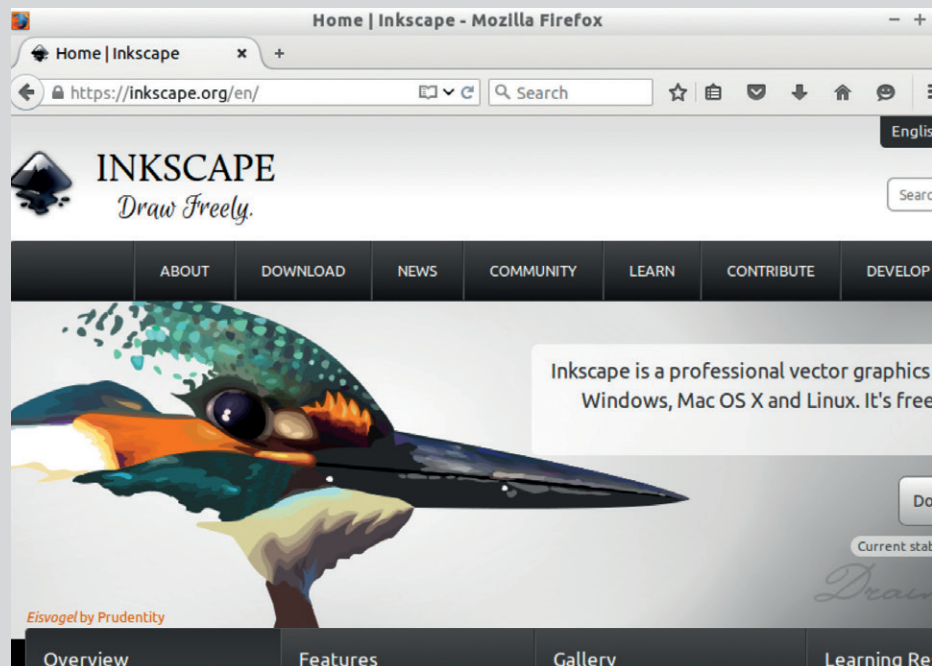
What we'd like to see, though, is a concerted effort to get more people involved in the less trendy aspects of FOSS development – like the aforementioned documentation and testing jobs. How about a non-profit Linux Documentation Team supported by donations from companies and the wider FOSS community, which funds efforts to improve documentation across Free Software projects? This is no silver bullet, but arguably it'd be a much more effective way to improve documentation than the current state where one person occasionally contributes docs to KDE, another to *GCC*, and so forth. Centralising and focusing efforts with a well organised funding drive could make a significant difference.

Another thing we'd like to see is more consolidation. Wouldn't it be great if two groups that worked on extremely similar projects considered merging them? That's easier said than done, of course, and there are often personalities and politics at play. At FOSDEM in 2014, we asked the developers of Mageia and OpenMandriva – two very similar follow-up distros to the once-famous Mandriva – if they would consider merging. There was lots of umming-and-ahing, references to minor differences between the distros, but not much enthusiasm for the idea.

Vision Thing

Again, we can't tell people what to do. But there are so many projects out there that exist due to “not invented here” syndrome, or petty squabbles, that could easily be brought together for the benefit of us all. Perhaps we need some kind of FOSS Dispute Arbitration body that takes an objective look at two projects that have split due to non-technical reasons, and that can bring them together without hurting the egos of those involved.

Something else that needs to be unified is version numbers. This is another topic on which everyone has their own opinion, but it would help the FOSS ecosystem if we standardised on the meaning of each number in a release. Our proposal, which is already used by many projects, would be like this:



- **2.0.0** Major release with significant new features or codebase changes.
- **2.1.0** Minor release with small feature changes.
- **2.1.1** Point release, exactly the same as with 2.1.0 but with only bug/security fixes

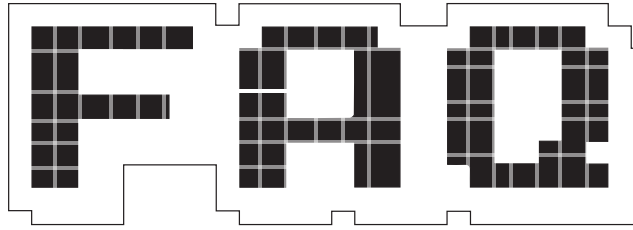
Some projects are guilty of using point releases to squeeze in new features or interface changes, which is just wrong. Users – especially in enterprises – should be confident that a point release won't break any workflow or require documentation changes, but merely fix issues. It's sensible, consistent, and enables big companies to keep software up to date without fear of breakages or random changes.

And finally, let's get some FOSS apps that are already widely used to version 1.0! It's really frustrating that *Inkscape*, one of the best Free Software apps out there, is still at 0.91 despite being developed for around 15 years. *Inkscape* is feature-rich, stable and polished, so release 1.0 already and tell the world! Yes, it's not up to us to tell the *Inkscape* team what to do, but a 1.0 release would be much more attractive to non-FOSS-insiders, especially those on Windows or Mac OS X.

So that's how we see the FOSS world improving. We're in a good state, but there's still a lot to be done. But we'll leave our final words to those wisest of sages, Bill and Ted: “Be excellent to each other.” 📺

Inkscape is polished, featureful and already used by many professionals – so why has it spent over a decade at version 0.x?

There are many projects that exist due to “not invented here” syndrome, or petty squabbles, that could easily be brought together



Zephyr

The Linux Foundation has a new project under its control, and it's one that doesn't actually include any 'Linux'.

GRAHAM MORRISON

Q Those of us born in the grim 1960s and 1970s of industrial Britain will already know what Zephyr is – a car built by Ford that was designed to look like a sullen sci-fi robot on wheels. Surely a FAQ is unnecessary?

A We remember! The Zephyr was like the Ford Anglia's eccentric aunt! But no, unfortunately, this is the wrong kind of magazine to take that image further. Zephyr is just one of those words that has been used for countless things, from a light westerly wind to the codename of a Soviet spy. Quite why the Linux Foundation chose it for its latest project is out of our search scope.

Q The Linux Foundation! So, there's a link to Linux then?

A Yes! It's the latest project to be announced by the Linux Foundation, the trade association that promotes the interests of its members

at the big enterprise level, all of which have some interest in the success of Linux, from Google (gold member) and SUSE (gold member), to IBM (platinum member), Intel (platinum member) and Oracle (platinum member). Incidentally, it costs \$500,000 to become a platinum member, and \$100,000 to become a gold member. Linux Voice has got some way to go before we can afford either of those.

Q Doesn't the Linux Foundation have a habit of creating projects like this?

A As far as we know, it's never created a project named after a British car before, but it's certainly chased plenty of emerging technologies by creating collaborative projects in their honour.

Q Does that mean this project is about another 'emergent technology'?

A Yes. If we asked you to guess we're certain you'd say 'internet of the things.' And you would be correct. "Zephyr Project is a small, scaleable

real-time operating system for use on resource-constrained systems supporting multiple architectures," according to the publicity. The Zephyr Project is the Linux Foundation's attempt to grab a little of the zeitgeist.

Q Sorry, what is this Internet of Things thing again?

A You know how it seems almost every device now has an IP address – your watch, your fridge, your mobile, your set-top-box, your television, your central heating, your car, your smoke alarm, your music player? These are the 'things', and it's predicted that over the next few years, we'll have far more connected things. No one is quite sure what this future of IoT dominance will look like, but lots of experts agree that it's going to be huge. If you've got an idea, buy the domain name now.

Q What does the Internet of Things have to do with the Linux Foundation, and more specifically, the Zephyr Project?

A Linux is obviously going to be an integral part of this revolution for the same reasons it's been the central component in both the cloud and smartphone revolutions. It's open source, small, and secure. Anyone can build their own projects using Linux and put them into lots of tiny bits of

No one is quite sure what this future of Internet of Things dominance will look like, but lots of experts agree that it's going to be huge

hardware without having to buy a single software licence. But the Zephyr Project isn't Linux. It's built on a tiny kernel designed to run on tiny systems. The kernel and modules can run in just 8kB of memory, for example, which is half the space in an expanded ZX81!

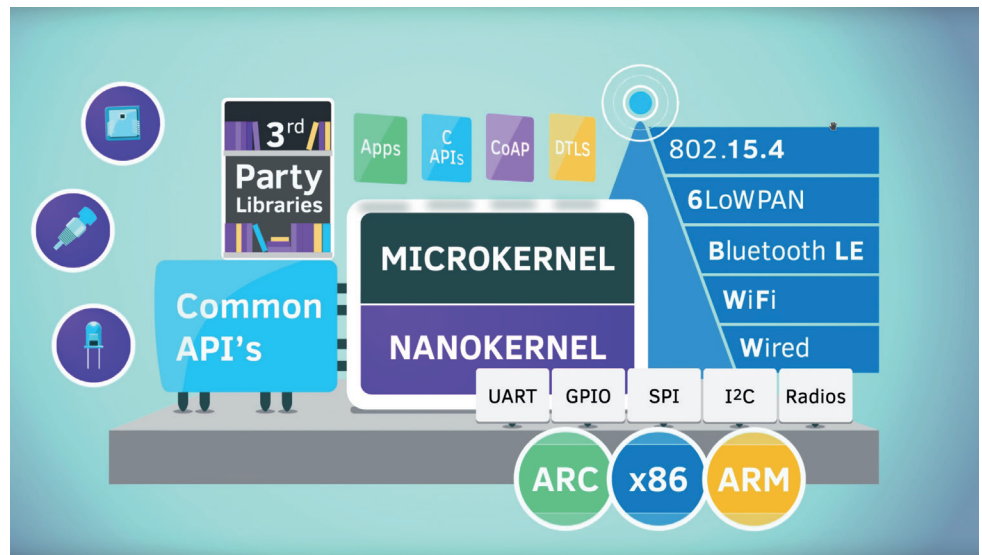
The kernel is the part of the operating system that turns inanimate hardware into something on which you can run applications – it's the Linux in GNU/Linux, and Zephyr is built to breathe life into those small things that need an IP address. Zephyr has also been designed to be a real-time operating system, which is something Linux finds hard. A real-time operating system is a serious endeavour, as they're often used in life-critical timing situations. They're designed to prioritise data above all else, such as feedback from a heart monitor or medical equipment. It's tempting to say Linux struggles with this because of PulseAudio, but that isn't true. Linux is just too complex.

Q What kind of devices does Zephyr run on?

A At the moment, the kernel runs only on specific boards, although that seems to have more to do with the project only just starting. It can generally run on x86 CPUs, ARM v7-M and v7E-M boards, and the ARC EM4 instruction set found on the Arduino 101. But if you wanted to play around with the project, the best way to do this would be on your normal PC or laptop using the *Qemu 2.7* emulator to run the kernel. You'll need to know what you're doing and augment the installation with your own code before it will do anything meaningful, but that's the idea behind the lot – connect small sensors and write bits of code to do something clever with the data.

Q Does the Zephyr kernel have anything in common with the Linux kernel?

A Other than being hosted by the foundation that employs Linus Torvalds, very little. This is perhaps best illustrated by the download size for both source code trees. The **tar.gz 1.0.0** release of the Zephyr kernel from March 2016 is a 6.8MB download. The equivalent March 4.4.4 release of the Linux kernel is 83.3MB. We dare not think about number of lines of code.



The Zephyr Project is a real-time operating system for tiny devices that provides APIs and connectivity to help you build tiny connected devices (Image source: zephyrproject.org).

Q Does that mean there's no Linux source code in Zephyr?

A When you look through Zephyr's source code, most of the files are attributed to Wind River Systems, Inc, which is itself a subsidiary of Intel (Linux Foundation platinum member). Wind River specialises in operating systems for small devices and has already developed its own commercial version of Zephyr, something it's calling the 'Wind River Rocket' OS. Zephyr itself has been released using the terms of the Apache 2.0 licence, which is considered only asymmetrically compatible with the GPLv2 licence used by the majority of the Linux kernel. That means the two projects are unlikely to share the same code, although with such a difference in emphasis, that would seem unlikely anyway.

Q Why would Zephyr be hosted by the Linux Foundation?

A Our cynical selves might say that launching Zephyr as a collaborative project under the auspice of the Linux Foundation is an attempt to bring a platinum member's project into the realm of wider industry exposure and adoption. But the Linux Foundation always seems to want a project a project for every facet of technological development. There's Dronecode, for example, an open source platform for Unmanned Aerial Vehicles (UAVs). There's Automotive Grade Linux for cars, and there's the

Cloud Native Foundation, which aims to "drive the adoption of a new set of common container technologies informed by technical merit and end user value, and inspired by internet-scale computing."

The Linux Foundation's current emphasis is on spreading the "collaborative DNA of Linux to other industries, companies and other projects," according to the Foundation's Executive Directory, Jim Zemlin, and we can see how that might include real-time operating systems and small kernels, even when there's no Linux link.

Q Where can we find out more about the project?

A Considering it's only just been announced, there's already a great deal of information about Zephyr, and we highly recommend you take a look at the excellent documentation that goes along with the project (<https://www.zephyrproject.org>). In particular, the part of the documentation that takes you through the application development workflow is very useful. If you have some coding skills, we're fairly sure you'll be able to build something, even without any prior experience in micro- and nano-kernels.

Q Anything else you'd like to add?

A Did we mention that it costs \$500,000 to become a platinum member and \$100,000 to become a gold member of the Linux Foundation? 📄

ALAN POPE

LUNCH WITH MR CONVERGENCE

Graham Morrison drives out to the Saxon settlement of Ferneberga to meet Ubuntu community manager and fellow Linux long-suffering podcaster.

It's been a year since the launch of the Ubuntu Phone – the launch where you could actually start buying the devices. Since then, Canonical, the company behind both Ubuntu and the phone's operating system, has taken a very community–

driven and organic approach to its growth, slowly introducing new models and new partnerships while supporting the hackers and developers who want to play, whether that's a port to a new device or building new applications. Alan Pope is at the

heart of this as a community manager at Canonical, but you may also know him from such podcasts as the long running Ubuntu Podcast, of which he is co-founder and co-host. We didn't need much of an excuse to meet up for a chat and a coffee.

LV Your role at Canonical seems quite multi-faceted.

Alan Pope: I work in the community team, and that means that I work with a bunch of other community managers. Canonical has a number of community managers in the community team and we have a manager of the community team – David Planella, and then there's myself, Mike Hall, Daniel Hallback, Michael Hall, Nicholas Skaggs and David Callé, and we all have our areas of focus. But mine is mostly core apps on the phone, but we spread ourselves a little bit away from whatever our area of skills is. Daniel is mostly working on Snappy at the moment, and David Callé mostly works on documentation, but we all pull together and organise things like events, or do Q and As and stuff like that. We each have a specialist area but we go beyond that.

LV Does the phone community, for example, consist of ordinary users or perhaps companies?

AP: I think of our users as extraordinary [much laughter!].

LV But after being involved in the community side of Ubuntu for such a long time, starting with—we'd imagine user groups discussing the desktop and the operating system – that must have changed?

AP: It has, and in the early days it was a

very exciting place to be, and I think similar to many user groups where you had enthusiasts and new people and people asking technical questions, getting together and doing demos – all that kind of stuff.

There was a lot of that in the early days of Ubuntu and there were a lot of people contributing to the platform – people who were packaging software that goes into the archive, or maybe doing translations for stuff that goes into the archive, or doing documentation or writing How Tos. There was a lot of that in the early days, and I think a lot of groups – not just Ubuntu – a lot of communities seemed to have moved on from that 'Getting the Platform Done' to other roles.

LV We've had the year of Linux on the desktop!

AP: Yep! Job done [laughter], we're all finished now, let's move on. I don't know exactly what it is. I know a lot of communities are seeing drift. Years ago if you went to a tech conference, something like FOSDEM or SCALE, or any other tech conference that has a Linuxy-bent to it, it was all Thinkpads and Dells, and you knew everyone in that room was running some version of Linux and that they'd probably had to edit their **Xorg.conf** to make it work. Or use *Ndiswrapper*, or some wacky stuff like that.

Nowadays, if you look around at FOSDEM there are a lot of (Apple) Macs, some of which are running Linux. A lot of them are not. The people who are running Linux, either on those Mac or on the die-hard Thinkpads, they've probably only needed to put a USB key in and it worked. Or if they were really lucky, they went out and bought it and it works. So the motivation for you to go to a user group and help someone edit their **Xorg.conf** is much lower, because that stuff is a solved problem.

That doesn't mean there aren't other problems for us to solve; it's just that the lower-level stuff is a lot easier now. You can go and get a USB key and put it into almost any computer – any distro – and it just installs. We, as in Ubuntu, and lots of other communities atrophied our community a little bit, because they've moved on to more interesting things. Maybe that's

You can put a USB key in almost any computer and it just installs

hardware hacking and they're into IoT and Raspberry Pi, or something else that's more interesting than what they used to do, or maybe they've moved on to the Mac because it was too much faff to get Linux working. I don't know



exactly where those people have gone but I miss them!

LV **The computing trend has also moved away from the computer and laptop and towards smartphones and mobile computing, you've followed them in your role at Canonical.**

AP: Yes, totally, and so I now install Ubuntu on a machine and then use it as a tool. This is one reason why I can't get my head around people who are constantly distro hopping. There is a level of learning you get from trying different distros, sure, but I just can't get my head around people who are like, "This icon is in the wrong place – I'll wipe the distro and start again." I use the Linux desktop as a tool to do my job and so the desktop is not particularly interesting to me any more, and I think that's probably true of a lot of people. It's not super-interesting any more.

LV **Was there a time when the desktop was super-interesting to you?**

AP: Totally! When I started contributing to Ubuntu (2005–2006), I was working where I had a lot of downtime, and I would spend a huge amount of time answering support questions online. We have a thing on [Canonical's] Launchpad called 'Launchpad Answers.' And I would go there on a daily basis and just look for the new and

unanswered questions, and sit there and answer people's questions. I had no idea who these people were, or where in the world they were, and they asked all kinds of things like "How do I get my wireless working?"

LV **Hence your memory for Ndiswrapper!**

AP: Exactly! It's still raw in my head. And that documentation we wrote all those years ago for *Ndiswrapper* is no longer applicable because wireless now just works, generally.

I remember one woman who asked a question that her husband had a piano and liked to make music and wanted to get this particular piece of music software working on Ubuntu, and so I went off and installed it on my machine and fiddled around and got it working, and then replied to the question – going back and forth a few times. And I know that question is archived back in the mists of time in Launchpad somewhere, and it was really refreshing to talk to brand-new users about something that they needed help with and I could help with just a few lines on a page. But now we get people asking, "How do I do this on Ubuntu Phone," and yes, that is where the interesting stuff for me is now.

People are saying, "How do I make my app do this?" or "How can I make my app run forever," or "How do I do SSH on my phone," which are all cool

and interesting things that most people on their phones don't do. It's quite good fun to do that.

LV **So there's the same category of 'hackers/users', it's just now they're aligned to more modern technology rather than the desktop or getting Linux to work?**

AP: I think so. And a lot of those people might use Ubuntu or they might not. We get people on our mailing lists who use Arch Linux, who want to develop apps for Ubuntu, or maybe they use Fedora and they bought an Ubuntu phone just to see what it's like. And it's

We get a community of interested people coming along for the ride

early days, and there are things that don't work yet – it's not quite finished.

LV **Is it every going to be finished?**

AP: Is anything ever finished! There is no version 'n' where there is not a number higher than 'n'. I've yet to see anything finished in the software world, so I think we'll continue iterating. But it's fun when I get people sending me screenshots of mockups or whatever of an app they've created, or they've created a new design for an existing app. I want to grab hold of those people and point them in the right direction and then show them the tutorials for getting started on Launchpad, or show them where the source code is.

I had that just this week. Someone pinged me on *Telegram* and said, "I've had an Ubuntu Phone for over a year now, and I'm quite interested in getting involved in developing the core apps." Someone has come looking to contribute, and sure enough I pointed him in the right direction and now he's contributing to our calendar app. It's stuff like that that people can easily get started on on Ubuntu Phone.

LV **Do you think part of the motivation behind the Ubuntu Phone was to keep that community of people who want to contribute themselves, as well as getting the platform on Chinese OEMs, for example?**



Where the desktop was, the phone is now, says Alan, stroking the rich mahogany Louis XV escrioire.



Alan used to use Red Hat, then moved to Debian when a bloke he knows told him Debian was “quite cool”. Well done that man!

AP: I think it's more a by-product of what we're doing. While Android is open source and Apple's iOS is not, they're still very similar in the way that they're released – there's a lot of development that's done internally and then at a big conference somewhere, the 'thing' is announced with great fanfare. A short while after, in the case of Google, it's thrown over the wall and there is your new software.

We aren't doing it like that, very deliberately, which means that people see the warts and all as we go along. If we'd closed the doors, brought up the shutters, and said we're not going to show you anything for another year, people would see a big change in what had happened between last year and this year. Whereas we're iterating over a year so that people see the individual changes, and people blog about an icon change, or they blog about the colour of something changing, or they're interested in a bug that affects them. Whereas that doesn't tend to happen in Apple Land or Google Land.

We get a community of interested people coming along for the ride as we're doing this development. I think it's a by-product of what we're doing – the fact that we're doing it in the open,

rather than a core goal. We do want to have a community and we want to have people interested, and without those people interested, you know, 'tech people' who are willing to write documents on GitHub about how to manage your phone or create apps, then we'd be in a difficult place.

Half of the apps that ship on the phone, by default, were written by people in the community, not by people at Canonical. Without those people, where would we be? We wouldn't have a calendar, a clock, a music app, a document viewer! Those things are important to us. And similarly, those people in the community then contribute to the core as well. There are people who have contributed to the web browser – recently the browser had a UI revamp and one of the contributors of that, a student living in Italy called Ricardo, he helped implement the new UI on our web browser. The community is very important to us and those people are enthusiastic. They come and go, we've noticed, over the years but we're still there for them when they come back.

LV Ten years ago, when you were helping people on Launchpad,

you weren't working for Canonical, so what led you to Ubuntu?

AP: I was having problems with my Philips webcam that meant I had to keep re-compiling the driver every time I updated Debian's kernel. That frustrated me, and my friend from my local LUG (Hugo!) said, “You should have a look at this Ubuntu thing, it looks quite cool.” I downloaded Ubuntu late 2004, early 2005 and started using it and thought it was great. I've never really had to really compile the kernel since then.

That was a milestone because I never had to go through that faff just to get a thing working. Obviously I still compile kernels now and again for various reasons, but I've not had to just for a webcam. But the thing that attracted me more was, yes, the community. There seemed like a really positive vibe around Ubuntu while at the time I found there was a bit of negativity around Debian – you could get chastised for asking the wrong question in an IRC channel, or you could get berated on a mailing list for saying the wrong thing the wrong way and that disappointed me.

Some people loved Debian – they loved the rigidity of Debian's rules, and that's fine, but I didn't really like that. I

like the look of Ubuntu and people seemed good fun.

I was contributing in the community for a long time; up until 2001/12, I was in various leadership positions in the Ubuntu community, and then I took a step back and thought, "Am I doing the right thing here?" I was doing my paid job in the day and this volunteer stuff in the evening – I'm doing a lot of volunteer stuff and I was thinking I needed to consolidate. I stepped down from a few of the positions.

Did you intend to leave the community completely?

AP: I kind of did. I resigned from a couple of positions and I actually got a personal message on IRC from Mark Shuttleworth [Founder of Canonical/Ubuntu] asking me whether there was any background story, or whether there was any reason why I was leaving because when people quit *en masse*, sometimes there is a back story, and there may be a reason and you might want to find out more, or whether it's fixable. He messaged me and asked if there was any reason and I said I just need to focus, really, because I'm spreading myself too thin. At that point, he said there might be a role for your focus at Canonical. Some month or two later, a job came up and I joined.

I had actually applied for jobs at Canonical a couple of times before, but I was unsuccessful, and Canonical do have a history of hiring from the community. There are a lot of people who work at Canonical who previously were students who worked on Ubuntu packages, for example, and were hired in. There are a lot of people in Canonical like that and so I was one of those. Sometimes people say that the Ubuntu community is shrinking – that's partly because Canonical is hiring them all!

Do you still feel just as excited by the phone?

AP: I'm not a big fan of Android. I'm not a massive fan of iOS. I especially don't like the whole going and downloading a random ROM from some obscurely named guy on the internet and flashing that on my phone because it will give me better battery life. Or I get another ROM that gives me better access controls on my phone and having to seek out these phones. I realise there's

a massive community around that, and people love doing that and people love hacking, but I don't distro hop. I want to have a thing that I can hack on that I don't need to keep flip flopping all over the place to be happy. On my desktop, I've done that and I'm on Ubuntu and I'm happy there. On my phone, I quite like the fact that on Ubuntu Phone I don't have to install all these random ROMs and I can hack on my phone, because I like being able to hack on my phone and do all kinds of cool interesting things. Just being able to SSH into my phones is quite cool.

The launch of Ubuntu Phone had a big impact on those ROM communities.

AP: I think possibly we should have engaged with the ROM community, if we want to call it that, perhaps a bit later. I think we engaged a bit too early because the sand was shifting a bit at the time and so a lot of people saw what we had on the phone and then they took that and ported it to other devices. And then various technical decisions we made changed the way the platform worked, and so none of those images work any more.

Things like the Nexus 10 and Nexus 5?

AP: That's slightly different because we only had one person maintaining that. It would be great if we had a pool of people. But what happens in the Android community seems to me that you have one guy who makes his ROM and then someone else forks that and makes his ROM, rather than multiple people work together on one really good one. There are groups that have emerged from that – like Cyanogen – but it would be great if we could have groups of people working together.

There's a guy called Marius who did a port to the OnePlus One and a couple of other devices, and he's now working on the Fairphone 2, and he has a whole bunch of other phones on his list that he'd love to get to, but he's just one guy. Whereas in the Android community one person is focused on getting a ROM working on their one device because you're scratching an itch – you're trying to get it to work on that device because that's the one you have. What we would rather do is have multiple people

working on one device to make that a really good ROM, and then move on to the next one, so we end up with a line of devices that are of a similar state. Bluetooth works on all of them, Wi-Fi works on all of them, accelerated hardware works on all of them, but it's difficult when that's quite a specialist task, porting Ubuntu to a phone.

Is the community getting larger?

AP: I think there are a few things in this platform that are missing or that are

We've been banging the convergence drum for a long time now

not fully fleshed out or fully finished, that have bugs in them. And I think as time goes on and we fix those, and also that we, Canonical, have agreements with partners with more devices, it becomes more interesting to people. When there's only one device on the planet, it's a lot harder for people to get interested, and it's also harder when those devices aren't available in every country. There's huge demand in the US, for example, where we don't have a good story for a phone you can use over there yet. In the future we will, I'm sure of that.

When you look at the number of people Google has to throw around at Android, or that Samsung has to throw at Tizen, and we're a 700-people company – that's the entire company, not just the people who work on Ubuntu Phone. People who work on Ubuntu Phone are in the tens, it's not hundreds, so it's an upward struggle and we are punching above our weight. You look at the size of our stand at Mobile World Congress [perhaps the biggest mobile event in the calendar] it was humongous, showing off community ports, community code. One of the best demos I'm told was the music app, because we've recently created a converge version of the music app so that it works on tablets, phones and when you rotate it it behaves differently. And that was one of the better demos given at MWC. The demo was written by an Oxford student in the UK. It wasn't created by us. We

If you'd like to see a port of Ubuntu on the Fairphone 2, or indeed any other device, you can go to <https://ubports.com> to find details of how you can get involved.



gave him some design input but he created that in his spare time, and those people who are willing to create something cool and interesting to demo to other people will start getting other people in, as well as the porting community as well.

LV Was convergence always intended to be part of the Ubuntu Phone operating system?

AP: It's funny because I was clearing out my old email last night and I did a search for everything older than the beginning of 2016 to archive off my Gmail, and it showed up some emails from 2011 and I saw an email entitled, "Convergence Demo" and "Convergence devices" and stuff like that, and that was five years ago, so we've been banging the drum for a long time. But it's only now coming to the point where you can plug your keyboard and mouse into a device or Bluetooth attachment, and use it like that. It's taken a long time to get here and there have been a couple of diversions long the way.

LV Does much of what goes into Ubuntu Phone get rolled back

into Ubuntu?

AP: If we find a bug in Network Manager or PulseAudio, the packages that are on the phone are the same – it's the same package as from the (Ubuntu) archive. If you do a `dpkg -I` on the phone, you'll see Network Manager and PulseAudio.

There was a fun bug we had maybe eight months to a year ago. We couldn't figure out why the phone would sometimes suddenly start screeching really really loudly. I had this at a recital concert at my daughters school and just out of the blue my phone just started screaming at me! A real high-pitched wailing noise and it took a lot of digging to figure out that it was a PulseAudio bug. It was the fact that PulseAudio was never really designed to be on mobile devices like this.


LV Was it ever designed to be on a desktop though?

AP: No comment! I love PulseAudio!

LV What do you like about PulseAudio!

AP: It just works. For no other reason than in the same way when I moved

from Debian to Ubuntu I no longer have to compile my webcam driver. Now, with PulseAudio, I can have *Skype* open – well maybe not now! – *Mumble*, I can be playing a Flash video, I can get notification sounds from my browser. And so on the phone with the bug we had, it turns out that when the phone gets a notification it makes a 'ting' sound and then the phone goes back to sleep. Then you get another notification and it goes back to sleep again. When the phone goes 'ting' the first time, PulseAudio is playing that noise through the sound card.

Then the phone goes back to sleep but PulseAudio carries on sending the zeros to the soundcard because it anticipates there's more audio coming down the line for a period of time. The CPU goes to sleep and then you get another notification – the CPU wakes up and starts sending zeros to the soundcard and the soundcard freaks out and you get this screeching noise. That bug was fixed and that fix will ripple down to the desktop in the same way because it's the same package, and it's the same with *Network Manager* and *GStreamer*. 



LISTEN TO THE PODCAST

LINUXVOICE

WWW.LINUXVOICE.COM



BUY LINUXVOICE MUGS AND T-SHIRTS!



shop.linuxvoice.com

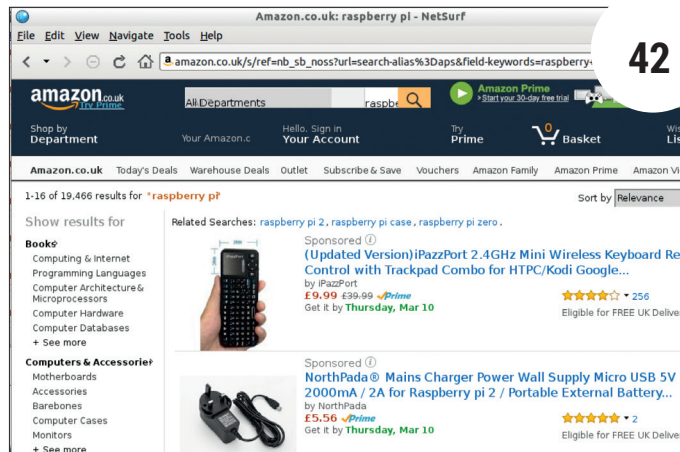
REVIEWS

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



Andrew Gregory
Needs some spare time in which to play with his Flotilla kit from Pimoroni.

On test this issue ...



42 NetSurf 3.4

With all the new features it keeps adding, *Firefox* is at risk of evolving itself out of existence. If you agree, try *NetSurf*: it's small, fast and could be the One True web browser to take back the web.

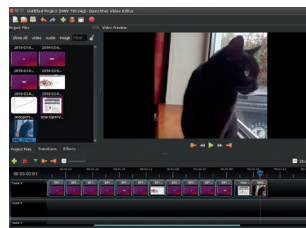
Spring is in the air, which means a few things. First, there's a fresh release of Ubuntu on the cards, reminding me that I really must get round to putting Ubuntu on my smashed (again) Nexus 5 smartphone.

Second, there are many, many things to be done. Last month the Linux Voice team launched a Kickstarter campaign to get a children's coding book published. We failed to reach our target, but the idea is still a great one. We need more coding materials, more volunteers at Code Clubs, more online resources. It's the future, and with Beep Beep Yarr! we want to help make it happen. It'll take a little longer, but we'll get there, because it's worth doing.

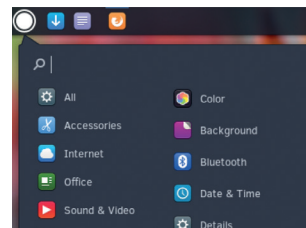
Brown rain

Spring is when projects arise, blinking from their winter slumber. The allotment needs work if we're going to get the potatoes planted in time. The weather station needs building, websites need to be coded, words need to be written, businesses launched. And, as always, Free Software makes it possible. Now let's get stuck in!

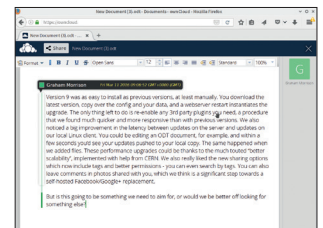
andrew@linuxvoice.com



43 OpenShot 2.0
In the already packed world of video editing, can this crowdfunded offering stand out?



44 Solus 1.1
Arising from the ashes of SolusOS, Solus looks like a good bet for the desktop user.



45 Owncloud server 9.0
The workhorse of any cloud infrastructure, Owncloud gets better with every release.

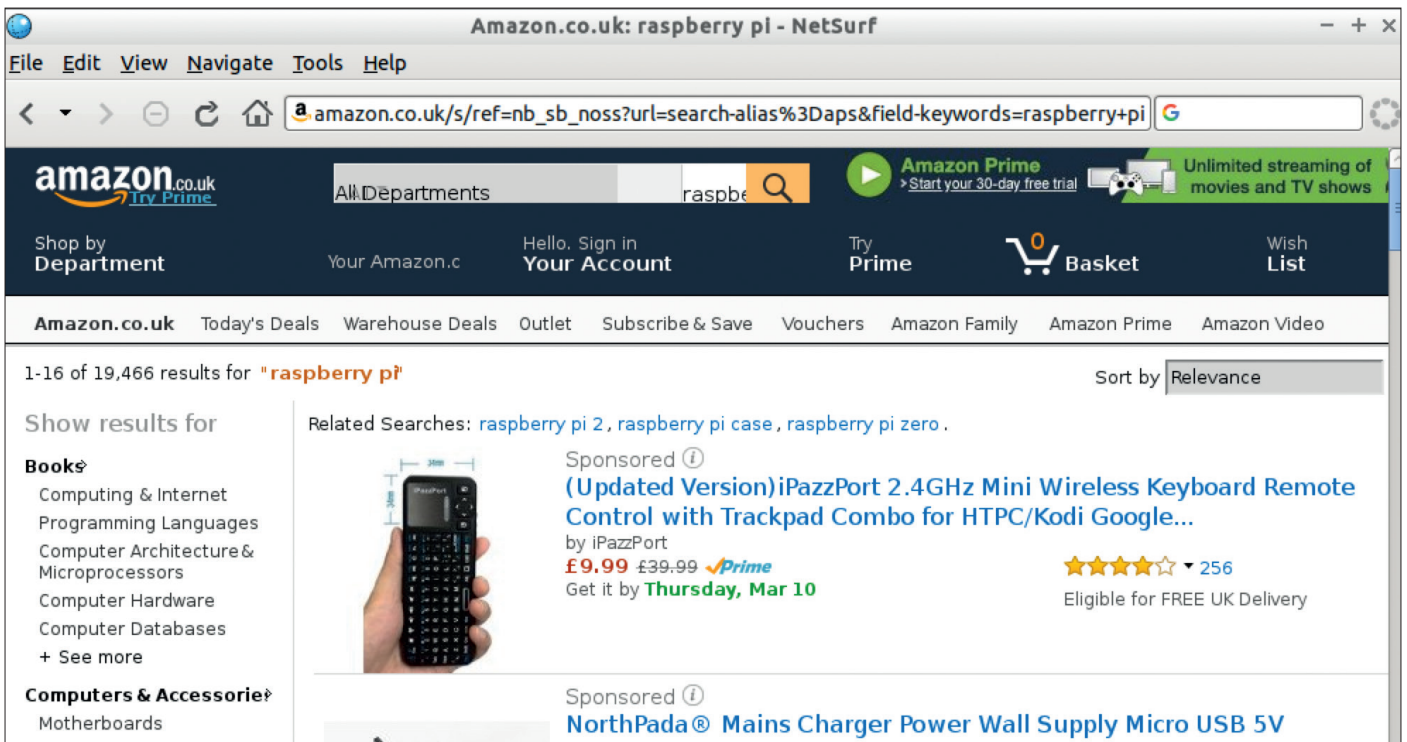
Group test and books



48 Boooooooooooooooooooooo!!!!
If you learn how to program games, you've learned how to program anything – at least, that's the theory. We just like shooting things.



50 Group test – privacy distros
The tools are out there to ensure your privacy, but why install them yourself when someone else has made these user-friendly distros for you?



NetSurf 3.4

After getting frustrated with Firefox, Mike Saunders needs a fast, light web browser.

Web www.netsurf-browser.org
Platforms Linux, Unix, RISC OS, Haiku, AmigaOS, Atari
License GPL v2

Firefox has always been one of our favourite FOSS projects (and in fairness, it's still very good), but recent changes to the interface, coupled with the addition of unwanted features like the Pocket integration for saving offline content, have started to annoy us. So while it's still our browser of choice for most cases, we're always on the lookout for something else. And *NetSurf* has been on our radar for a while. Unlike most browsers doing the rounds, it doesn't use an existing HTML rendering engine like *WebKit*, *Blink* or *Gecko* – but instead has its own, built from scratch.

NetSurf is designed to be highly portable and usable on old machines running relatively obscure OSes. So if you're still battling on with an old Amiga or Atari, or you're rocking with a Haiku OS or RISC OS installation, then *NetSurf* provides a fairly robust browser. It's also scorchingly fast and friendly with your RAM banks: whereas a freshly started *Firefox* munches up 221MB

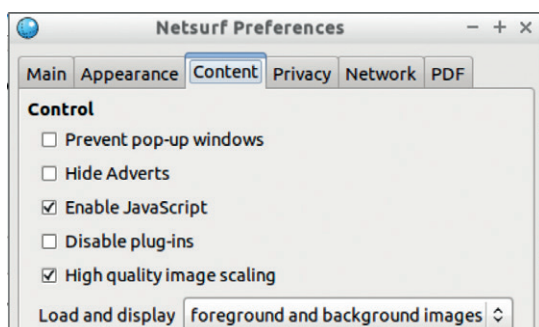
viewing the Wikipedia page for Linux, *NetSurf* only needs 76MB. Multiply that over many tabs and you save a lot of memory.

In addition, *NetSurf* offers most of the basics you expect in a browser: bookmarks, cookie management, pop-up and advert blocking, Do Not Track, and even a smattering of JavaScript support. In this release, JavaScript is provided by the Duktape engine, but the results are a mixed bag due to a limited set of bindings. Some sites are fairly functional, whereas others struggle or just don't work at all.

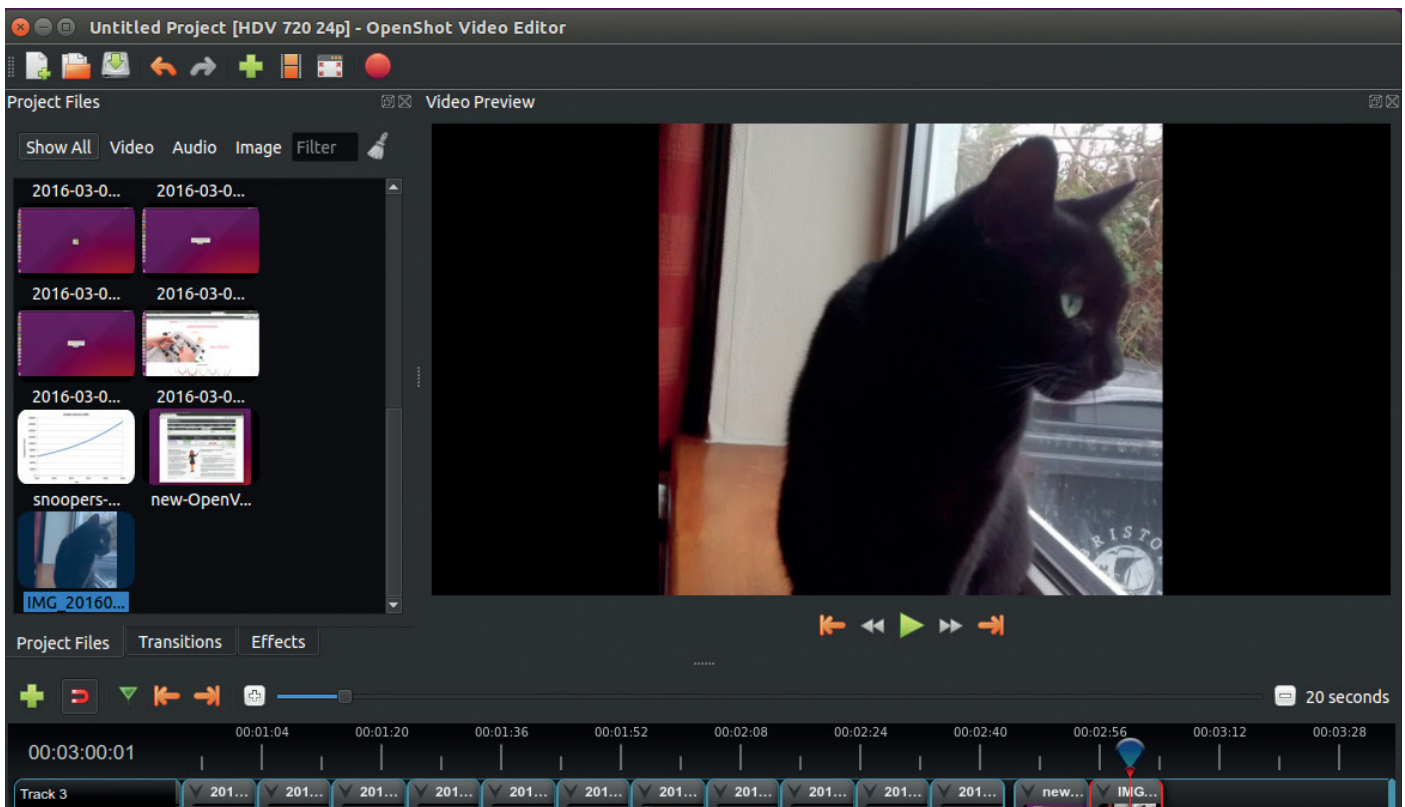
NetSurf has come on in leaps and bounds from the 3.3 release we tried last year, and works well for casual browsing of simple sites, but it still gets confused in places. Of course, creating a HTML and CSS rendering engine that covers the ever-growing specification is a mammoth task, and all of the developers of *NetSurf* are working on it out of passion rather than as a paid job, so the achievements they've made already are to be applauded.

What we say is: definitely give *NetSurf* a try. It won't work flawlessly on every site you visit, but for those times when you just want to spend a bit of time on Wikipedia, Reddit and other text-oriented sites, you'll find it fast, sleek, and very pleasant to use.

Making great progress, and usable on plenty of not-too-demanding websites, but still glitchy in places.



JavaScript support may be disabled when you first start *NetSurf* – to enable it, go to Edit > Preferences and the Content tab.



OpenShot 2.0

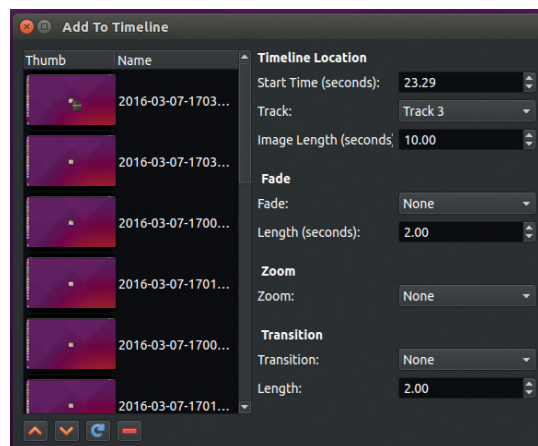
Ben Everard needs to edit his many cat videos, can OpenShot enhance the cuteness?

On 17 April 2013, The OpenShot project raised \$45,028 in a crowdfunding campaign to develop version 2.0 of this video editing software with a target to release in December 2013. Development slipped, and the final version didn't come out until early 2016. Version 2.0 represents almost three years of work to just about every area: the rendering engine has undergone a major rewrite, the interface has moved from *GTK* to *Qt*, and the software now runs on OS X and Windows as well as Linux. Hopefully this will bring in a larger pool of users which in turn will mean more people to find bugs and develop patches.

There are only a couple of new features in *OpenShot 2.0* – most of the work has been tidying up the existing features rather than adding new ones. The Split Clip and Timeline dialogs do make it easier to construct videos, the former being used to manipulate long videos and the latter used for constructing videos out of a large number of short clips.


Our biggest complaint with earlier versions of *OpenShot* was the stability. During our editing with *OpenShot 1.x*, we found that crashes were part of the normal workflow. *OpenShot 2.0* did crash during our testing, but nowhere near as frequently.

Had *OpenShot 2.0* met its delivery target of December 2013, it would have been without a doubt the best video editor on Linux at the time.



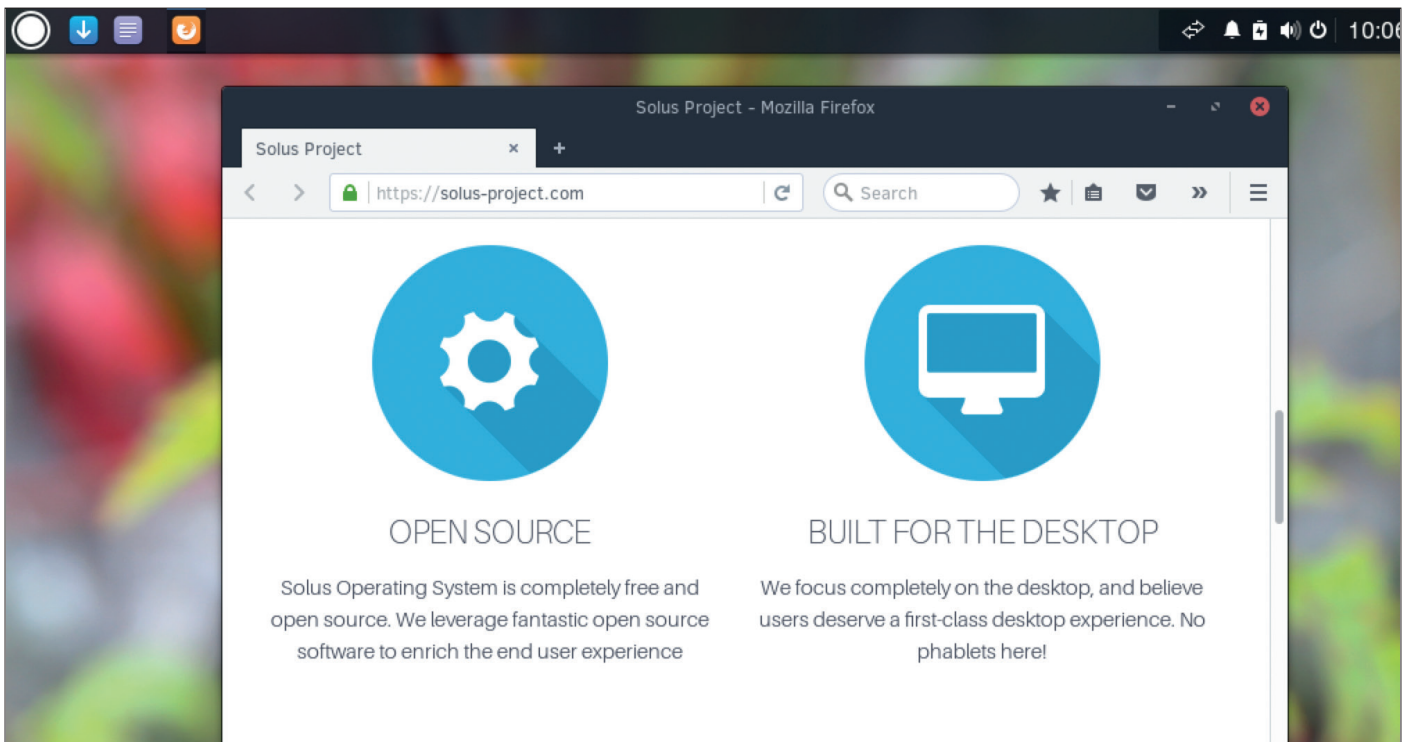
Web www.openshotvideo.com
Developer Jonathon Thomas and contributors
Licence GPL v3

The new Timeline dialog is perfect for making video slideshows from a collection of photos.

The problem is that the last three years have seen a renaissance in open source video editing. *Pitivi* has also had a significant funding drive, *Lightworks* on Linux is now considered stable, and *Kdenlive* continues to improve. In this competitive landscape, the new and revamped *OpenShot* can still hold its own, particularly with people who only do occasional video editing, because the interface is engaging and easy to use. 

A great interface pushes *OpenShot* to the front of the increasingly competitive field of Linux video editors.





Solus 1.1

SolusOS is dead – long live Solus! **Mike Saunders** examines this desktop distro.

Web <https://solus-project.com>
Platforms x86-84
Price Free


You may recall SolusOS, a newbie-oriented Debian-based distro that used the Gnome 2 desktop with a bunch of custom add-ons. SolusOS was abandoned in late 2013 due to a lack of development manpower, but the project has recently been revived under the name Solus. Now, given that there are more desktop-focused Linux distributions in existence than there are atoms in the universe, why are we covering Solus here?

For starters, it's extremely polished. So many distros simply take Gnome, KDE or Xfce, chuck on a custom theme, and leave it at that. Solus goes a lot further by incorporating its own written-from-scratch desktop environment, Budgie, which is now available in some other distros such as Arch. Solus "focuses on simplicity and elegance", with a sleek program menu and notification centre – but fortunately it's not a victim of "not invented here" syndrome. Budgie

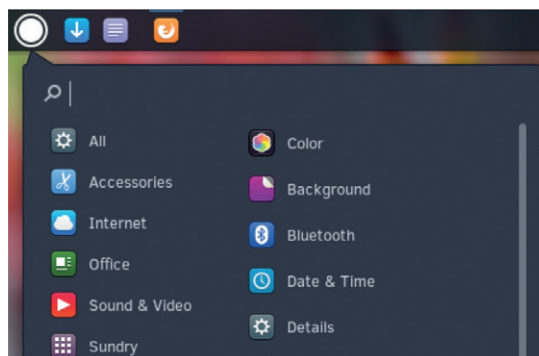
uses some components of Gnome, so it's somewhat analogous to Cinnamon on Linux Mint.

While Budgie looks lovely, it has its limitations. Launcher icons in the top panel, for instance, don't have tooltips, so you can't be sure what they do without clicking on them. If you want to install Solus from live mode, you need to click the blue down arrow button – but that same button is also shared by the Software (package) Installer in the menu. Similarly, there's little in the way of customisation to be done – forget about context menus for the panel or many of the notification area icons.

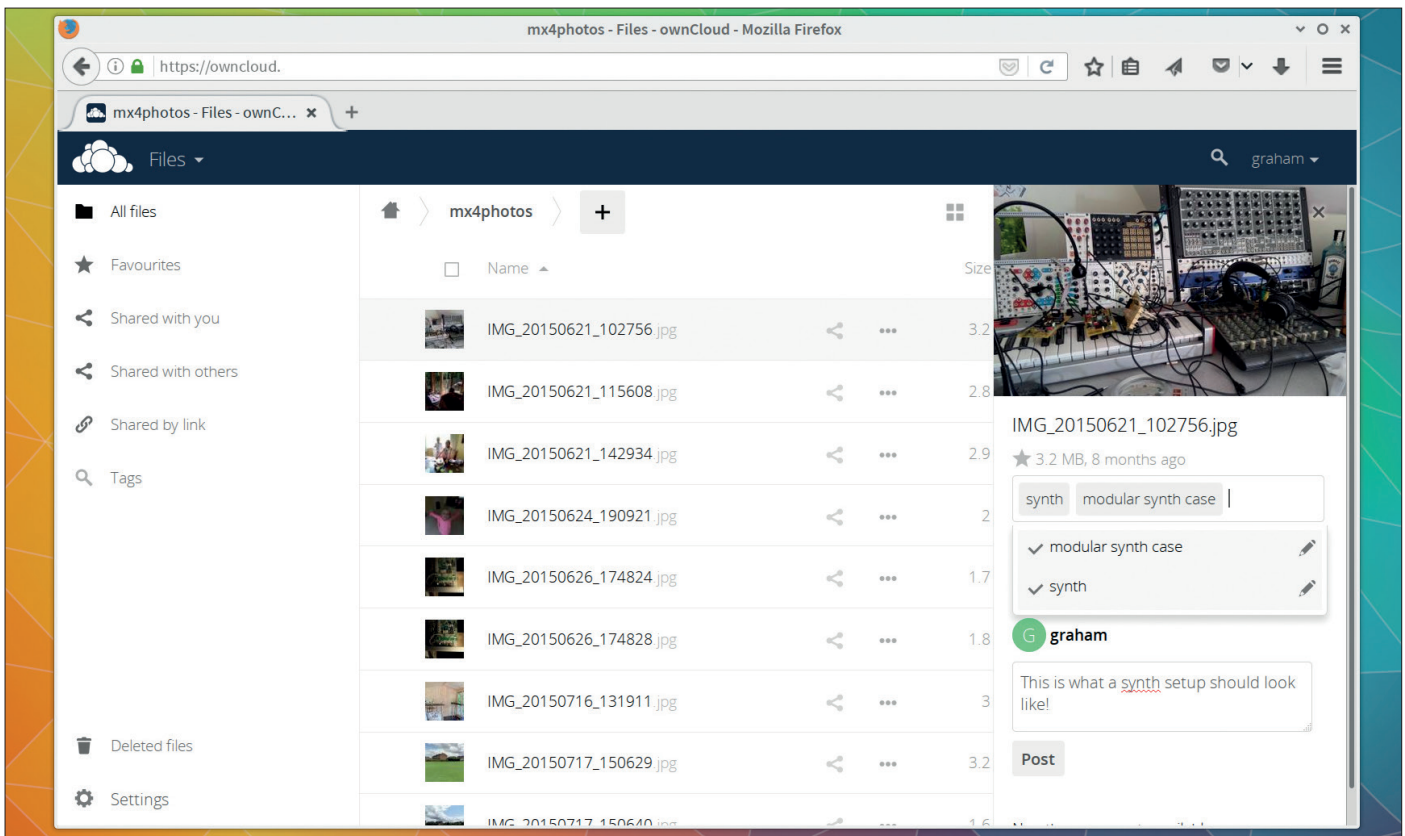
Otherwise, Solus provides a handful of FOSS desktop apps: *Firefox*, *Thunderbird*, *VLC*, *Rhythmbox* and various Gnome utilities. We were surprised to see no office software installed by default; most users would expect that, and it's not like the distro is trying to squeeze onto a single CD.

In all, we like Solus for its attention to detail and focus on good presentation (remember – this is a feature), from the website through to the desktop itself. Finding the right balance of usability for both newbies and more experienced users is going to be tough, but with a bit more work, it has the opportunity to be the Ubuntu or Mint of its generation. 

An ambitious distro that could lead the pack when it comes to the desktop, but needs more refining and a better out-of-the-box experience.



The main menu is attractive and well-designed – it's just a shame the distro doesn't include more apps as standard.

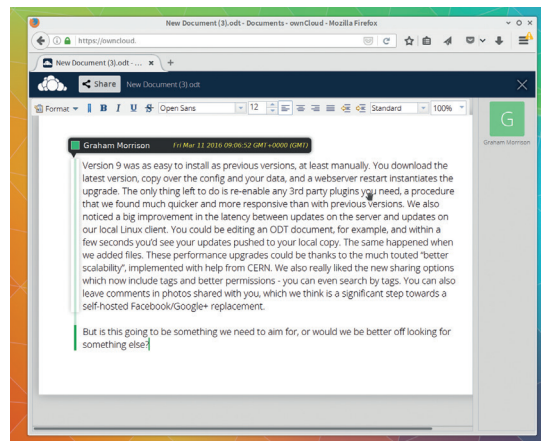


OwnCloud Server 9.0

Graham Morrison takes another great leap away from Facebook and Google.

The progress *OwnCloud* is making is almost unbelievable. Not only is the project going from one major release to the next, cramming great new features into each update, its user base has grown from 2.4 million in 2015 to 8 million in 2016. This is completely unprecedented, but also perhaps understandable. It's a slick, professional, beautiful data repository for all your files. It synchronises with your desktop and phone, adds context via calendar, contacts and even document editing plugins, and can of course be self-hosted. As we've mentioned before, we use *OwnCloud* extensively to put the magazine together and it has never failed us.

Version 9 was as easy to install as previous versions, at least manually. You download the latest version, copy over the config and your data, and a web server restart instantiates the upgrade. The only thing left to do is re-enable any third-party plugins you need, a procedure that we found much quicker and more responsive than with previous versions. We also noticed a big improvement in the latency between updates on the server and updates on our local Linux client. You could be editing an ODT document, for example, and within a few seconds you'd see your updates pushed to your local copy. The same happened when we added files. These performance upgrades could be thanks to the much touted "better



Web <https://owncloud.org>
 Developer OwnCloud Inc.
 Licence AGPLv3

You can now edit documents collaboratively, and it works well even on low-end servers.

scalability", implemented with help from CERN. We also really liked the new sharing options which now include tags and better permissions – you can even search by tags. You can also leave comments in photos shared with you, which we think is a significant step towards a self-hosted Facebook/Google+ replacement, as is the cooperative document editing with annotations. It's all brilliant. 🍌

The rapid user-base expansion and new features make this release stable and full of collaboration.



GAMING ON LINUX

The tastiest brain candy to relax those tired neurons



RUN. JUMP. SHOOT.



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

It's been anticipated for a long time now, but OpenGL's successor, the Vulkan API, has finally been released in what promises to be a huge leap forward for Linux and cross-platform gaming as a whole. Along with the release, Nvidia has provided drivers with Vulkan support. There are no drivers currently released on the AMD side (despite the company having developed the Mantle API on which Vulkan is based), though their release is imminent.

The cherry on the cake is that this release comes with an update of *The Talos Principle*: the first Linux game to support Vulkan. Current performance is not game changing, but given the beta state of the drivers and a game which was built around DirectX 11, it is very encouraging in this early stage. There is also already talk of *DOTA 2* being shifted to Vulkan, as well as a few others.

Currently, DirectX 12 is only supported on Windows 10, which accounts for around 13% of desktop operating system usage. With Vulkan supporting myriad operating systems and versions, this alone provides an obvious and significant advantage over the proprietary alternative and no doubt will prove to be attractive and cost-efficient for developers looking to publish across platforms. With developers hopefully opting to develop solely on Vulkan and not on DirectX with a subsequent OpenGL port, this should mean more games for us at similar performance to those on Windows, if common sense can overcome the network effect.

XCOM 2

A worthy successor to one of the best strategy games out there

Website <http://store.steampowered.com/app/268500>
Price £34.99

XCOM: *Enemy Unknown* was hugely successful, so expectations for its sequel were extremely high. Thankfully, these expectations have been met and often exceeded. Firstly, there are no drastic changes from its predecessor, and given that it received universal praise, that's probably a good thing. The most fundamental difference is the change of circumstances, where instead of spearheading a trans-national defence team against alien invasion, it now focuses on a much leaner rebel team working to overthrow the alien regime after its successful invasion. This provides a far greater sense of urgency and resourcefulness to a game in which humanity was already the underdog.

XCOM 2 strikes a careful balance between making the game familiar enough so as not to anger fans with drastic changes, but not subtle enough that it feels like a lazy upgrade of its predecessor. It does a great job in this regard,



Missing that 90% accuracy shot reminds you that *XCOM 2* is as unforgiving as ever.

and it does feel like a new experience with continuity from where the previous game left off.

Combat remains the least changed aspect, with the base-management experiencing the most changes. One criticised change is that some missions now have turn limits, though this does add a sense of pace and urgency to parts of the game previously lacking it. The turn-based tactical combat and the management aspects of the game provide that much-appreciated variety and detail, all held together by a fun plot and kept interesting through steady progression.



Wonky close-ups and shooting through walls still occur, but still aren't that annoying.

XCOM 2 feels like a new experience with continuity from the previous game

Firewatch

A gripping mystery that really hits the mark

Website <http://store.steampowered.com/app/383870>
Price £14.99

Every so often a game comes along that shows the true merit of video games as an artistic medium.

Firewatch is one of these.

The game immediately sucks the player in with a harrowing backstory of a Henry and his wife Julia. Within minutes, Henry (the protagonist) finds himself traversing the Wyoming wilderness to arrive at a lone firewatching tower where he will

spend the summer. There are no other physical characters in the game, and the only human contact is a relationship which develops through the walkie talkie to another firewatcher named Delilah. Without giving too much away, the plot revolves around the mysterious and sinister goings on in the surrounding area, while Henry comes to terms with his unsettling past.

Firewatch is recommended to just about anyone who enjoys a good story and doesn't crave constant action.



Firewatch's graphical style manages to be both beautiful and light on resources.

American Truck Simulator

The game that shouldn't be great, but is

Website <http://store.steampowered.com/app/270880/>
Price £14.99

"Simulator" is often a euphemism for "niche", "parody" or just "terrible", but the *Truck Simulator* games have been there to remind us that this isn't always the case. This latest instalment is no exception and brings trucking to its spiritual home in the United States. It's time to put Johnny Cash on the truck radio and watch the diners and motels pass by on the way to San Francisco.

How a game series in which one drives sensibly at the legal speed limit for long stretches has become so popular defies conventional logic, but to those who have played this or its predecessors, it makes perfect sense. Playing *American Truck Simulator*, for a lot of people, is a great way to unwind and listen to music, all the while



Like everything in the States, the trucks are flamboyantly big.

having the nice gaming mechanics of building a logistics company from scratch on top of the driving.

Thankfully, this isn't just a reskin of *Euro Truck Simulator 2*, adding plenty of content on top of the new locations and trucks. The cities are more fleshed-out than in its predecessor, with more buildings and landmarks, which only adds to what feels like a more genuine trucking experience.

ALSO RELEASED...



Dying Light: The Following DLC content is often not worth mentioning, but *The Following* is so massive it could easily be a standalone game. Additions include a huge new map, drivable vehicle, new weapons, a new story and more. Like the base game, the story is poor at best, but it does extremely well in providing endless entertainment to an already very entertaining game.
<http://store.steampowered.com/app/239140>



Agatha Christie – The ABC Murders The crime and mystery genres tend to work rather well with point-and-click mechanics, and it's great to see such a game based on literature. The game has some lovely visuals, good voice acting, nice music and captures the essence of the novels. Graphically, it does suffer from some aliasing issues, even on highest settings, but this can be forgiven if you're after a solid "whodunnit" story.
<http://store.steampowered.com/app/374900>



Superhot Re-imagining the FPS genre, *Superhot* is a game in which time only advances as the player moves, which makes things very interesting indeed. Add in some stylised graphics, take away mechanics like regenerating health and ammo drops and you have a game that's proved to be insanely popular. Now you can play a shooter in which you can have a coffee and file a tax return between killing enemies.
<http://store.steampowered.com/app/322500>

Learn Game Programming With Ruby

Ben Everard knows that creating a game is just the first step to being beaten at it.

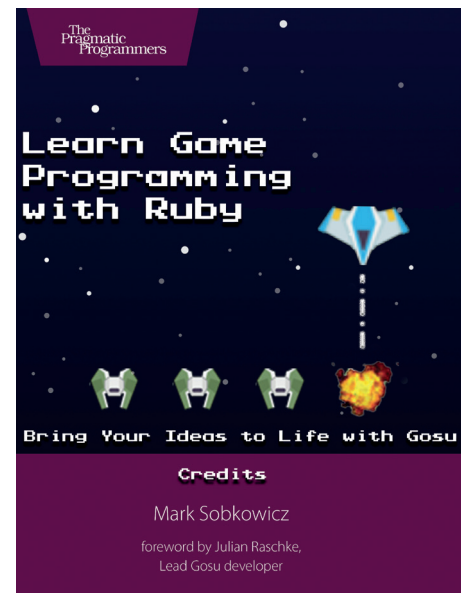
Author Mark Sobkowicz
Publisher Pragmatic Bookshelf
Price £15.99
ISBN 978-1680500738

Before turning to a career writing about open source software, this reviewer had a brief stint teaching English as a foreign language. It taught me exactly one thing about getting students to learn a new subject, and that is that by far the most important thing is to get your students interested in the topic. If they're excited to learn about something, it doesn't really matter how badly you convey the information. *Learn Game Programming with Ruby* is really a book about learning object oriented programming, but it uses games to get new programmers excited about the subject.

Rather than use a games library, Mark Sobkowicz builds all the games using just a media library and shows the reader how

to construct the internals of the game from scratch. This makes it more of a book about programming than a book about learning one particular games development framework. *Learn Games Programming With Ruby* starts a little too quick for someone who's not programmed before, but it makes a great second book to get new programmers thinking a little more about the best ways to structure their programs after they've mastered the basics. The book's getting started guide only covers OS X and Windows, but all the libraries used are cross platform and work on Linux.

A fun way to build up your programming skills.



Learn Game Programming With Ruby focuses on 2D games such as vertical scrolling space shooters. Fine by us!

Secure Your Node.js Web Application

Ben Everard secures web apps with an axe – cut the network cable and you're safe.

Author Karl Duuna
Publisher Pragmatic Bookshelf
Price £23.99
ISBN 978-1680500851

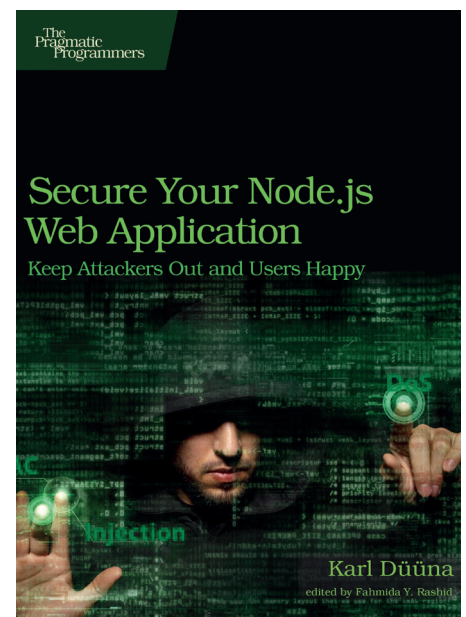
Security is vital for web applications – it's a subject we've covered many times in Linux Voice. Most of the older web technologies – such as PHP – are already well known from a security perspective, and there are many resources to help developers know what mitigations to take to ensure that their software remains secure. Node.js is much newer, and this is the first book to really go in to the security implications of writing sever-side code in JavaScript (which is a very different prospect to writing secure client-side JavaScript).

The basic techniques used to attack Node.js are the same as those used to attack other server-side technology: code injection, authentication-bypass, cross-site scripting, etc. However, JavaScript has

some peculiarities that make it vulnerable to these attacks in unusual ways, and if you're developing Node.js then you need to know what to avoid to stay secure.

Secure Your Node.js Web Application doesn't go into much depth about the different types of attack, and ideally we'd like to see some examples that aren't trivial and contrived. Software security is a complex topic, and security problems are often subtle and hard to spot. This book will help readers avoid most basic mistakes, but it's not detailed enough to help developers write really secure code.

Not as in depth as we'd like, but currently the best option available.



Apparently, attacking a website requires a transparent touchscreen and a hood to obscure your face.

OpenStack Cloud Computing

Graham Morrison just can't get his head out of the clouds.

Author Kevin Jackson, Cody Bunch, Egle Sigler
Publisher Packt Publishing
Price £31.99
ISBN 978-1782174783

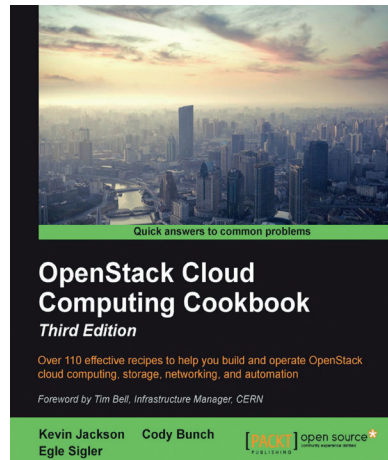
Learning about OpenStack can be a challenge. It's a complicated subject, and that complexity seems to be increasing each year, mirroring its growth. The main problem is its scale, both in terms of what there is to learn and its requirements. It's not a simple case of installing a few packages or experimenting within the confines of a virtual machine, and that makes learning about OpenStack and playing with new ideas difficult.

We discovered this while writing our own tutorial on getting started with OpenStack (see p76), and while it is possible to install the system on a single relatively powerful machine, it negates the reason why OpenStack has been so successful – the ability to scale from just a few machines to thousands. And while OpenStack's documentation is very good, what we really need are some practical examples of various deployments.

Cooking with OpenStack

We didn't have the space in the title above, but this is the third edition of a the *OpenStack Cloud Computing Cookbook*. The 'cookbook' part is important, because OpenStack, in particular, is very modular, and each of those modules can be thought of a single ingredient. To this end, the book starts off looking at each module in turn – Keystone, Glance, Neutron, Nova, and Swift. Starting with Keystone, the authentication and identification module, says a great deal about the attitude of the writers, dealing with the vital and thorny subject of security first, before moving on to the more practical modules.

The book uses the Ubuntu Cloud Archive, which hosts the OpenStack installation packages for Ubuntu. It's the same installation of Ubuntu we used for our own tutorial, and so the code examples and the explanations



This is a great book if you've ever thought about trying OpenStack.

within the book should be easy to understand, and unlike with Canonical's own (excellent) installer, by performing each step yourself, you learn more. When you create the tables used by Keystone in SQL, for example, you know what those tables are doing and how they fit into the entire OpenStack ecosystem, slowly chipping away at its complexity.

With all the ingredients covered, and keeping with the cookbook metaphor, there are plenty of specific 'recipes' littered throughout the book. The cover mentions 110, and it does feel like almost every step gives an alternative example for each step. This is the most impressive part about the book, although you'll likely need a more explanatory book if you want to take your OpenStack installations further. Background information here is kept to a minimum so that you can rapidly progress from one example to another, while providing only the bare details of what each bit of text. But this level of detail is perfect for our short attention spans. There are some excellent explanatory diagrams littered throughout, and there's a 'How it works' section after covering each concept, but mainly, the book just teaches you how to get on with it.

An excellent way to get OpenStacked.

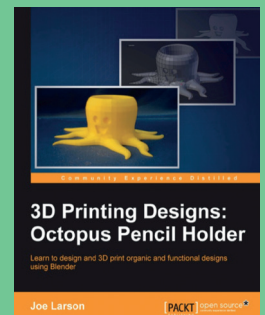


Also released...

April 2016

3D Printing Designs

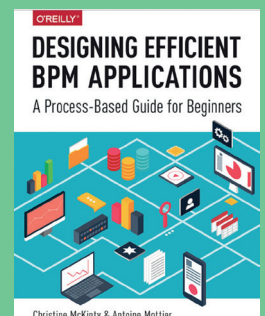
3D printing is still going from strength to strength and getting cheaper every year, but there aren't a great deal of quality examples to follow. This book focuses on creating an octopus pencil holder, which looks great from the cover, and should also get children excited too. This book promises to take you from *Blender* models to final object, step-by-step.



Everyone needs a good pencil holder.

Efficient BPM Applications

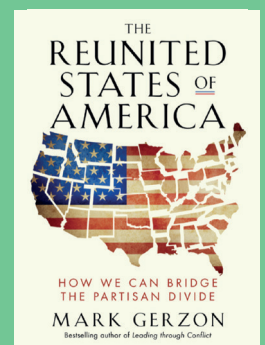
We were initially drawn to this because of the title – BPM to us means beats per minute, and we can think of lots of ideas for creating some lovely crunchy beats. But no, this isn't that BPM, it's apparently Business Process Management. If you already knew that and think we're crazy for mentioning beats, we're guessing this book is for you.



BPM != Business Process Management.

Reunited States of America

What perhaps shouldn't be surprising is that this book is being published by one of O'Reilly's partners, and could have some exposure in some of its more technical audiences. We'll avoid the politics, but we're always interested in ideas that are designed to bring people together rather than move them apart, which is exactly what this book promises to do. 🇺🇸



Coming to America.

GROUP TEST

Since Harry Potter's invisibility cloak is of little use on the internet, Mayank "Griffindor" Sharma checks out some distros that will do the job.

On test

IprediaOS



URL www.ipredia.org
Licence GPL and others
Latest release IprediaOS 1
A live distro built on the I2P anonymous network.

JonDo Live-DVD



URL www.anonymous-proxy-servers.net/en/jondo-live-cd.html
Licence GPL and others
Latest release 0.9.88.2
This distro offers multiple privacy tools.

Lightweight Portable Security



URL www.spi.dod.mil/lipose.htm
Licence GPL and others
Latest release 1.6.4
A privacy-centric distro from the US Department of Defence!

Tails



URL www.tails.boum.org
Licence GPLv3+
Latest release 2.0.1
The most popular distro for privacy seekers that anonymises via Tor.

Ubuntu Privacy Remix



URL www.privacy-cd.org
Licence GPL and others
Latest release 12.04r1
A modified Ubuntu distro for creating an isolated environment.

Whonix



URL www.whonix.org
Licence GPL and others
Latest release 12.0.0.3.2
Takes a unique virtualised approach to privacy.

Privacy distros

Until the Snowden leaks, the proponents of privacy attacks were digital outlaws firmly on the wrong side of the law. However, much to our chagrin, we suddenly found ourselves in the midst of an information age where invading our privacy has not only become a state-sponsored practice but has also become a mainstream business model.

Despite the ensuing hue and cry, apps and websites by vendors on the IT high street continue to gather information about us, usually without our consent. As users of open source software, the all-seeing eye of the developer community shields us from the nefarious practice of bundling backdoors and malicious code prevalent in closed source software. But we're

still exposed to all sorts of privacy leaks and security intrusions as soon as we go online. The mere act of surfing the web and using the various online file sharing services and social networking sends out more information to unintended recipients than you can imagine.

Over the next few pages we'll look at different Linux distros that take steps to ensure your anonymity and protect you against inadvertent leaks and breaches. All these distros are designed with privacy protection as their primary objective, but ensuring privacy often comes at the price of usability. Since the distros on test take different approaches to privacy protection, we'll be on the lookout for the distro that guarantees utmost safeguards with minimum disruptions to our way of working.

We're exposed to all sorts of privacy leaks and security intrusions as soon as we go online

Anonymity vs Privacy

Privacy and anonymity are two different concepts that are often confused with each other. While they are both becoming increasingly necessary as our online movements are constantly tracked, it's vital to understand the differences between them.

Privacy is the ability to keep some things to yourself; in contrast, anonymity is when you want people to see what you do, just not that it's you doing it. In terms of our digital existence, the most important distinction is that anonymity

is when you are non-identifiable whereas privacy is concerned with your ability to cut off yourself selectively.

To illustrate, if you visit a website via Tor, that website will not be aware of your real location, in essence concealing your identity. However, if you visit a site via Tor and then log into your account, you are no longer anonymous. But by routing your request via an encrypted network, you haven't revealed your destination to any intercepting party (such as your ISP), thus maintaining your privacy.

Security Qubed

From virtualisation to compartmentalisation.

Qubes OS is an interesting distro for security conscious users. The distro divides the computer into a series of virtual machines or domains using Xen that are cut-off from one another. Each virtual machine only has access to the services that it needs to perform the designated function, thus limiting the potential security threat. Despite all that virtualisation wizardry, Qubes offers a coherent and streamlined

desktop. The install-only distro is based on Fedora and allows the user to install either KDE, Xfce or both the desktop environments.

The architecture of Qubes is different from that of a chroot environment in which the systems share hardware resources such as network cards. Besides the minimal graphical desktop, everything else in Qubes including network cards and even disks, are separate virtual machines.

The VMs in Qubes are known as security domains or AppVMs. Each of these has a different level of security and are defined in a template. By default, Qubes installs with three domains – work, personal, and untrusted, though users can add more through Qubes VM Manager. The distro takes some getting used to and new users should read through its documentation to wrap their heads around the distro.

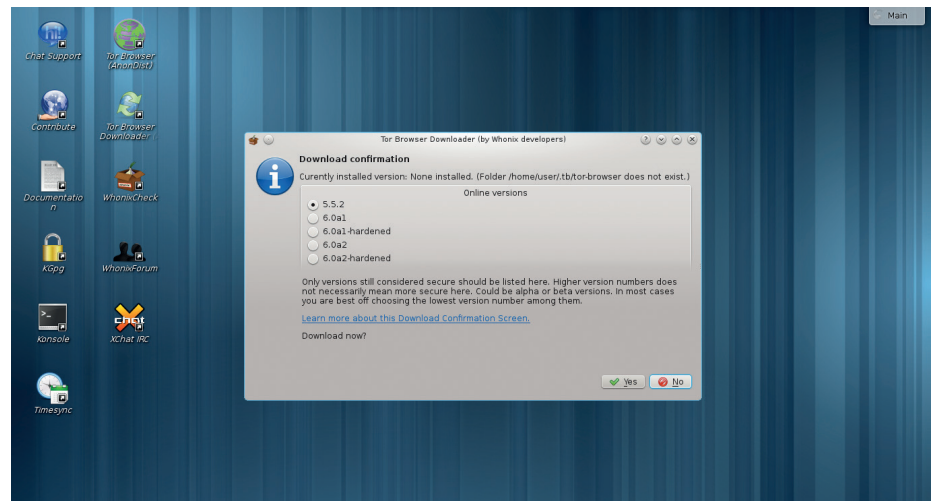
Whonix

Whole new world.

Whonix is very different from the other distros in this group test. The distro is in fact a pair of Debian-based virtual appliances that you must run simultaneously, in two separate virtual machines. The Whonix-Gateway is used for configuring Tor, and its only task is to route traffic via the Tor anonymising network. This machine has two virtual network interfaces – one connected to the internet via NAT, which is used to communicate with the Tor network, and the other network interface is connected to a virtual LAN. The other machine is the Whonix-Workstation, which is the desktop you are supposed to use for browsing and other tasks. The *iptables* rules on the Whonix-Workstation force it to only connect to the virtual internet LAN and redirect all traffic to the Whonix-Gateway. This scheme prevents apps from ever knowing the user's real IP address or accessing any information on the physical hardware.

At first launch both Whonix-Gateway and Whonix-Workstation take you through a brief setup wizard to familiarise you with the project and set up some components, such as the repository. If you have the resources, the Whonix developers suggest you run multiple instances of the Workstation VM, one for each task.

Whonix doesn't have very many apps but the ones it does are tuned for guaranteeing privacy. For example, the *IceDove* email client ships with the *Enigmail* extension for encrypting email. There's also *KGpg* for managing keys. The distro has an icon for



For an even more secure implementation, you can deploy the Gateway on the physical hardware of one machine and run the Workstation on virtual hardware inside a different host altogether.

the Tor browser but doesn't ship with one by default; instead the icon brings up a script to download one from a list of stable, new and hardened releases. There's also Xchat for IRC but surprisingly no app for instant messaging. Then there's the WhonixCheck app, which scans the current installation and tests the Tor connection.

Be aware that installing updates will take longer on Whonix than on other distros, as they are routed via the Tor network. While Tor frowns upon using the network for certain activities like downloading torrents, they have no issues with Whonix using the network for downloading updates.

Solid underpinnings

The distro is based on Debian 8 Jessie, whose repository you can use to install any additional packages. It's a good idea to refer to the documentation on the project's website before installing components like

the *Pidgin* IM, not just for the installation instructions but for some best practices to ensure anonymity and pointers to related additional privacy-enhancing components. In addition to information on the bundled components inside the distro, the Whonix wiki also makes for a good read for anyone interested in stopping inadvertent leaks.

There are several Whonix variants. You can download the Gateway and Workstation as *VirtualBox* images and import them on either Linux, Windows or Mac machines. Instead of using *VirtualBox*, Linux users are encouraged to instead download the two Whonix machines as KVM images.

You can also install Whonix on top of QubesOS mentioned in the box above.

VERDICT

A unique, albeit resource-intensive solution to the problem.

★★★★★

Ubuntu Privacy Remix

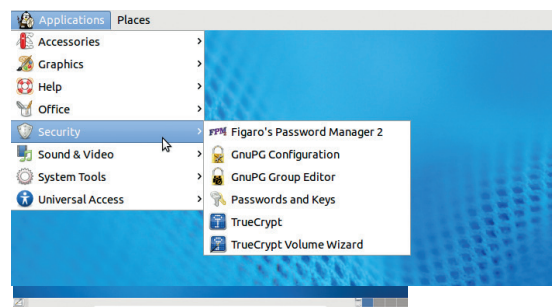
'buntu and privacy?

The two biggest threats to security and privacy are the internet and the data-sniffing trojans and scripts on your hard disk. The Ubuntu Privacy Remix (UPR) distro is a radical solution that works by cutting access to both of these. With UPR you get an isolated environment that can be used to work on sensitive documents or highly infected ones.

The developers have stripped support for all sorts of networking hardware from the distro's kernel, which has also been tweaked to ignore all ATA hardware such as hard disks. (You can still read files from optical drives and removable USB drives.) If you plug in a USB drive, you can create an encrypted TrueCrypt volume on the drive using the distro to save and read documents. UPR mounts all removable media as well as the TrueCrypt volumes with the **noexec** option, which prevents the execution of programs from the media.

Talking of TrueCrypt, UPR has something known as Extended TrueCrypt volumes. In addition to storing regular files the extended volume also stores the configuration and user data of *LibreOffice*, *Evolution* and *GnuPG*. This helps you overcome the disadvantage of losing your app settings that's most commonly associated with Live distros. Another noteworthy customisation is the distro's homebrewed front-end to *GnuPG*, which focusses on encrypting and decrypting files. To guard against cold boot attacks, the distro wipes the computer's RAM when you shut down.

The distro has all kinds of apps, from *LibreOffice* and *VirtualBox* to the *Gimp* image editor and the *VLC* media player. The latest UPR is based on the Ubuntu 12.04 LTS release and uses the Gnome 2 desktop. It also bundles detailed setup and getting started guides to orient new users.



Use the godmode boot option to gain superuser access with the `sudo su` command.

UPR is an ideal platform for anyone looking for a system that's truly off-the-grid and provides an isolated habitat that segregates the working environment from the files on the computer's disk.

VERDICT

A wonderful option for anyone looking for an isolated environment.

★★★★★

Lightweight Portable Security

Straight from the horse's mouth.

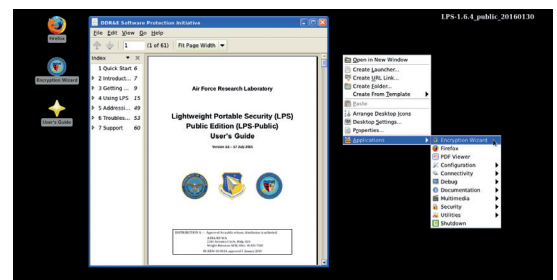
The Lightweight Portable Security (LPS) distro is created by the Software Protection Initiative (SPI) under the direction of the US Air Force Research Laboratory (AFRL) and the United States Department of Defense. They've designed it for telecommuters who need to access secure government networks from unsecured remote locations. To this end, the distro works with DoD-issued Common Access Card (CAC) and Personal Identity Verification (PIV) smartcards for accessing restricted government websites.

Just like the Ubuntu Privacy Remix, LPS also uses a modified kernel that prevents interactions with the local hard disk. The distro includes the public edition of the Encryption Wizard app created by the AFRL, which can encrypt and decrypt individual files and even complete directories. There's also an option to force the distro

to use DNSCrypt, which prevents DNS spoofing by authenticating communications between the computer and the DNS resolver.

LPS includes a handful of apps. There's the *Firefox* browser, which is equipped with the HTTPS Everywhere and User Agent-switching extensions. There's also a PDF reader and a bunch of remote desktop software including the *Citrix Receiver* and *VMware View*. These are complemented by a smattering of apps such as a barebones text editor and an image viewer. The SPI also produces a Deluxe edition of the distro which is the same as the regular release but additionally includes *LibreOffice* and *Adobe Reader*.

Although LPS uses the Xfce 4 desktop, it's been tweaked extensively to resemble Windows XP. Everything from the layout of the desktop, complete with the Windows-hallmark Start button, to the window



The goal of the developers is to put out new LPS release at least once every quarter.

decorations has been designed to ape the proprietary OS to give the DoD spooks the look and feel of a familiar environment. For further assistance, the project's websites contains thorough FAQs as well as a Quick Start guide and a user's manual.

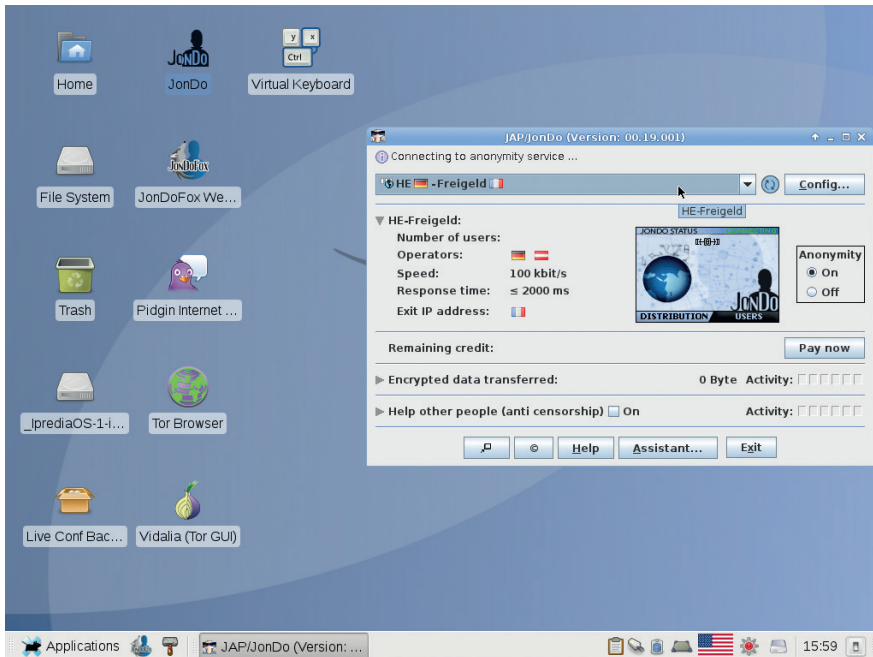
VERDICT

A privacy-focussed distro with some fascinating customisations and tools.

★★★★★

JonDo Live-DVD

Proximus prime.



You can use JonDo's controller to select from any of the listed mix cascades.

JonDo Live is a Debian-based distro built on tools that are designed specifically for masquerading your identity online. The core piece of software in the distro is the Java Anon Proxy (JAP) more commonly known as JonDo. The app anonymises your connections using a proxy service in order to conceal your origins. The Java-based client for the proxy is available for various platforms including Linux, BSD, Windows and Mac.

The live distro offers users the option of using the JonDo or the Tor proxy to anonymise your network traffic. JonDo routes your traffic via encrypted channels through a mix of multiple servers. While this routing ensures privacy and anonymity, hopping through the servers means your surfing will take noticeably longer. Although anyone can hop on the JonDo network via a free cascade, you can pay a small fee and instead use a premium mix that in return promises increased speed and better anonymity by routing traffic via more servers. Furthermore, the free mixes only allow traffic on port 80 and 443 (for HTTP and HTTPS). For everything else, such as 5222 for Instant Messaging and 21 for FTP, you need to subscribe to the premium mix.

Many of the included apps in the distro are preconfigured to use the JonDo

proxy for anonymity, including *Pidgin* and *Thunderbird*. There's also the privacy-enabled *JonDoFox* browser, which is configured to route traffic via the JonDo proxy and also includes a number of privacy and security themed browser add-ons. If you're using the Tor proxy instead, there's the *Vidalia* client for establishing and managing the connection to the Tor network as well as apps such as the Tor browser and *TorChat* designed to facilitate communication over the Tor network.

Something for everyone

Besides these network-specific apps, the distro also hosts several other privacy-focussed and general purpose apps such as *ZuluCrypt*, *KeePassX*, *Jitsi*, *Qtox*, *LibreOffice*, *Gimp*, *Skype*, *VLC* and more. To avoid setting up account information repeatedly on all these apps in a Live distro that can't be installed to the disk, the distro includes a nifty little app that rolls the configuration of several apps into a compressed tarball. Ferry the tarball via a USB disk and upon reboot extract in inside the home directory of the Live distro.

VERDICT

A good collection of anonymous apps for multiple proxy networks.

★★★★★

Be incognito on the go

Privacy on a stick.

Freepto is a Debian-based distro that's available as a live image designed to be transferred on to a USB disk. The distro includes a tool to create an encrypted persistent storage partition on the free space in the USB disk. Freepto is chock-full of all the popular digital invisibility tools as well as the popular desktop productivity tools to enable you to use the distro as your everyday desktop. There's everything from *LibreOffice* and *Gimp* to *VLC* and *Audacity*. For everything else, there's the *Synaptic* package manager and Debian's extensive repository of packages.

Unlike with regular distros, software like *Pidgin*, *Xchat* and *Filezilla* are configured to work over the Tor network. The distro includes the Tor browser and also a regular *Iceweasel* installation that's equipped with privacy-enhancing plugins such as Disconnect, Adblock Plus and HTTPS Everywhere.

Freepto also contains a set of tools for the privacy-conscious users. There's *BleachBit* for zapping temporary files, *Gufw* for tweaking the firewall, *TorTP* to force any app to route via Tor, *TrueCrypt* to work with encrypted storage, *MAT* for stripping metadata from files, and *tomb*, a command-line file encryption utility. The distro also includes a user's guide, but it's only available in Italian and Spanish. This shouldn't stop new users from experimenting with the distro, since it's pretty much like a standard desktop distro and fairly intuitive.



Freepto is built using the Debian Live Build tools by the hacktivist group Av.A.Na.

Tails vs IprediaOS

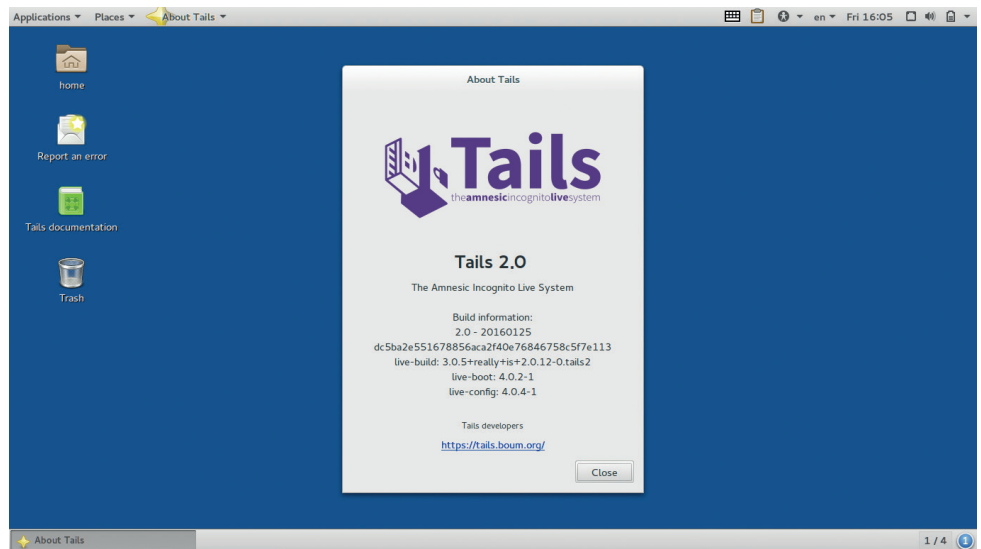
Which one does a better Keyser Soze?

The Debian-based Tails (The Amnesiac Incognito Live System) uses Tor to create an encrypted channel to the internet to preserve your privacy. You can install Tails to a USB disk and use the remaining free space to create an encrypted persistent partition to store settings and files.

Tails has the usual cocktail of apps that you'd find on a regular Linux distro, as well as some unique ones such as the *Electrum* Bitcoin wallet, the *Gobby* collaborative editor, *GtkHash* to generate and verify the checksums of transmitted files and *MAT* (Metadata Anonymisation Toolkit) for zapping metadata information from files.

Unlike many other distros however, Tails bundles the *Synaptic* package manager for fleshing out the desktop. This task requires admin access, which is disabled by default but can be set up using the *Tails Greeter* app. The app can also be used to spoof your device's MAC address to avoid it being used to uniquely identify you. On shutdown, Tails wipes the computer's RAM to safeguard against cold boot attacks.

Another anonymising network that's occasionally pitted against Tor is I2P (Invisible Internet Project). You can access the I2P from *Tails* by passing the **i2p** boot parameter when starting



Tails doesn't interact with a computer's hard disk, so nothing is saved to the computer you're running on.

the distro. Just like it does with Tor, the distro will connect to the I2P network in the background. When it's done, fire up the bundled *I2P Browser* in Tails, which takes you to I2P's browser-based control panel.

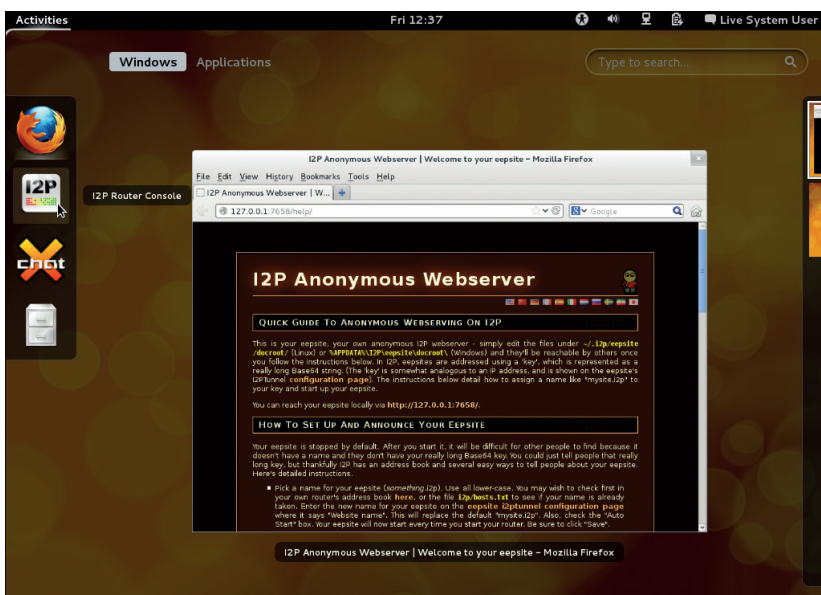
To P or not to P

IprediaOS is to the I2P network what Tails is to the Tor network. The distro can be used to connect to the I2P network and also includes some tools that can be used to access the various services on the I2P network. Perhaps

the biggest difference between I2P and *Tor* is that while *Tor* is used to anonymously browse the regular internet, the primary objective of I2P is to create and allow access to the hidden network of eepsites. Again for comparison, eepsites with the **.i2p** TLD are what **.onion** sites are to Tor.

The distro includes a ready-to-go webserver that's online as soon as you're connected to the I2P network. Besides websites, the network hosts various services such as anonymous email and anonymous file transfer. *I2PShark* is the name of the network's Bittorrent service, which can be accessed via the network's Python-based client called *Robert*. Sharing torrents is a popular activity on the I2P network, unlike *Tor*, which frowns upon the use of the network for this purpose.

IprediaOS uses the Gnome 3 desktop and is one of the few privacy-centric distros that can be easily installed to the hard disk. The distro however includes an older version of I2P, so your first order of business should be to update to the latest release before connecting to the network.



To get the most out of IprediaOS, users are encouraged to peruse its documentation.

VERDICT

TAILS LINUX
A well-packaged distro
Linux distro built
around Tor.

★★★★★

IPREDIAOS
Built around I2P but
hasn't been updated
for quite some time.

★★★★★

OUR VERDICT

Privacy distros

The requirements for digital privacy and anonymity are highly subjective. You might prefer a secluded computer cut off from the internet to work on and pass along encrypted documents. In such a case, your choice of distro will be very different from a whistleblower who wants to anonymously email a bunch of documents without revealing his name or location.

Given the US Government's track record when it comes to ensuring privacy it's difficult to suggest the Lightweight Portable Security distro to anyone, especially since there are far better alternatives on offer. Then there's IprediaOS, which loses out simply for being outdated. If you do need to use the I2P network, the Tails distro does a better job than IprediaOS. The Ubuntu Privacy Remix has a very limited use case. While its use of encrypted USB disks for securely ferrying files means it's not completely cut off from the rest of the world, the lack of internet access limits its use for everyday online tasks.

While we applaud Whonix's compartmentalised approach to the issues at hand, setting it up requires a fair amount of work. The distro comes with a learning curve that's steeper than some of the other solutions on offer and also requires greater system resources.

Tails is one of the best-known distros for anonymity and privacy. It's built on the Tor network, is regularly updated and equips you with the right tools for staying invisible both online and offline.

If you force us to pick one distro, it'll have to be JonDo, even though to make the most of the JonDo network, you'll have to subscribe to a premium mix. A JonDo network offers several advantages over Tor and I2P. For starters you can select your mixes. Secondly, although the servers don't keep logs, compromising one server in the mix will not reveal your identity, because of the architecture of the network. In case you're still not convinced, the distro can still be used to connect to the Tor network, so you win both ways.

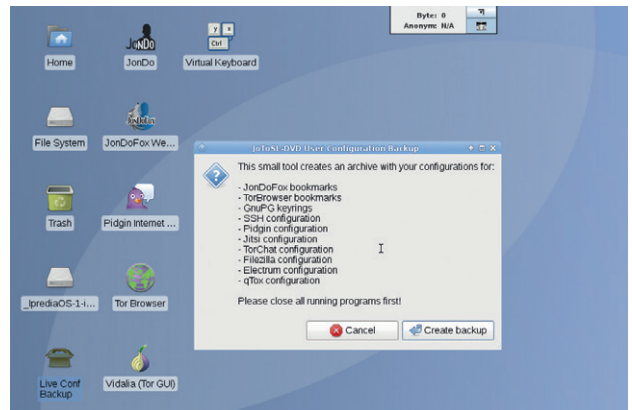
Because of their nature, the requirements for digital privacy and anonymity are highly subjective

The Freenet project

Freenet is a peer-to-peer system designed to facilitate anonymous sharing of information on the internet. The network enables users to communicate using a platform that cannot be censored. It creates a distributed data store that houses information accessible by all members. Freenet also publishes several tools for anonymous publishing and communication.

You can install Freenet on all desktop operating systems. Once it's installed follow its guide to create an anonymous identity using the WebOfTrust plugin. After creating an identity you can set up

a Freemail email address and configure your email client to send mail via this address. Similarly you can create an account on Freenet's Sone microblogging service, the FMS forums, and the FLIP IRC client. You can then interact with other members on the forums and even create your own website. Besides creating a secret identity, you can browse all kinds of websites on Freenet. The network maintains a list of Freenet websites with the most recently updated ones at the top. Since the content on Freenet isn't filtered, be prepared to encounter offensive and illegal content.



You can easily combine JonDo with other anonymisation services such as Tor.

1st JonDo Live-DVD

Killer feature: JonDo and Tor

www.anonymous-proxy-servers.net/en/jondo-live-cd.html

Experience the best of two anonymising networks and a host of relevant tools.

2nd Tails

Killer feature: Tor and I2P

www.tails.boum.org

If JonDo doesn't impress you, then use this.

3rd Whonix

Killer feature: Unique architecture.

www.whonix.org

A notable distro for the experienced privacy campaigns.

4th Ubuntu Privacy Remix

Killer feature: Disconnected and isolated.

www.privacy-cd.org

A truly standalone distro for powering a secure workstation.

5th IprediaOS

Killer feature: I2P

www.ipredia.org

An outdated distro for showing the I2P network.

6th Lightweight Portable Security

Killer feature: Encryption wizard

www.spi.dod.mil/lipose.htm

Doesn't offer anything uniquely different that's not available elsewhere.

Subscribe

shop.linuxvoice.com



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

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

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

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

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

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

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

NEXT MONTH IN LINUX VOICE

ON SALE
THURSDAY
28 APRIL



WHAT NEXT FOR UBUNTU

Once the world's favourite distro, Ubuntu has toned down the hype and got on with taking over – and it's coming to your TV!

EVEN MORE AWESOME!



Goldmembers

Steward of the Linux name, employer of some important kernel hackers – or corporate lackeys? Find out, as we go inside the Linux Foundation.



Microsoft's Linux

The end of the world is nigh! Microsoft has released a Linux distro! We take a look, while dodging fire, brimstone, plagues of locusts and other portents.



Rust never sleeps

It's been a year since Mozilla released the first stable version of its Rust programming language – high time we got our hands on it, we think.

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. Nothing in this magazine may be reproduced without permission of the editor, until December 2016 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2016
ISSN 2054-3778

Subscribe: [shop.linuxvoice.com](http://shop.linuxvoice.com/subscriptions@linuxvoice.com)
subscriptions@linuxvoice.com

FOSSpicks

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



Raised by endangered Tasmanian wolves, **Ben Everard** sniffs the air and howls at the latest and greatest Free Software releases.

Video player

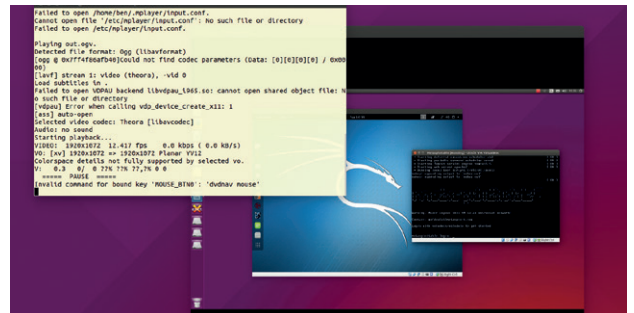
MPlayer 1.3

Version 1.3 of this veteran video player came out alongside the *FFmpeg 3.0* codecs, and the two work hand-in-hand. Because of this, the biggest update in *MPlayer 1.3* is that it now plays an even greater range of video files. *FFmpeg* handles the low-level details of what a particular file-format means, and *MPlayer* turns this into a useful player.

A quick `grep` through the *MPlayer* website tells us that there are 292

alternative front-ends for *MPlayer*. This tells us two things about *MPlayer*: that it's well used and that it's default interface is rubbish.

While *MPlayer* does have a GUI, it's more commonly used as a command line program. Its popularity comes from a phenomenal range of options that enable it to not only play just about any piece of video known to man, but also manipulate the video in any way you wish. This power does



MPlayer is the ultimate video Swiss Army Knife, but the command line options can be hard to fathom.

lead to it being a little confusing to use – hence the range of alternative front-ends.

PROJECT WEBSITE
www.mplayerhq.hu

Process monitor

htop 2.0

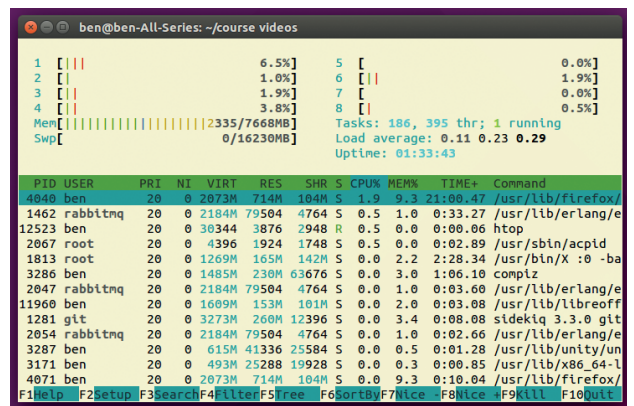
Linux is just a kernel – the heart of the OS on top of which everything else sits. The kernel has its own special place in memory and controls access to the CPU and other resources. Every other program that you run is a process, and it all starts with process 1, which initialises your system. Every server you run, every command you execute, every desktop application you open: they're all processes.

Process viewers help you keep an eye on all the different processes running on your machine, and **htop** is one of the best. The main job of **htop** is to give you an overview of what's going on inside your machine: which processes are using the CPU, which are eating up the memory, and how many free resources are there. It manages all

this through an *ncurses* terminal interface, which both looks good, and works well over SSH.

htop is the first program we turn to when investigating performance on our Linux boxes – desktops and servers. It doesn't always have all the information we need, but it does give a great overview of what's going on, and shows you where to drill down further.

If you're still using the bog-standard command **top** (the default process viewer on many Linux distros), then it's time to switch to **htop** and be amazed by the neater interface and mouse controls. **htop** has colours to provide a quick



Everything you need to know about your running processes in a colour-coded terminal screen.

overview of what's going on, better sorting of processes, and a tree view to give a better view of multi-process programs. With version 2.0, **htop** works on Linux, OS X and many BSDs, so you can now enjoy the same interface on all your Unix-y boxes.

PROJECT WEBSITE
<http://hisham.hm/htop>

It's time to switch to **htop** and be amazed by the neater interface and mouse controls

Windows-like OS

ReactOS 0.4

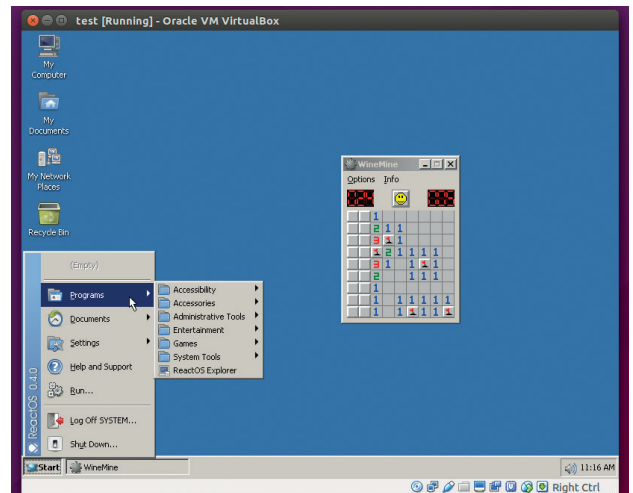
Version 0.3 of this Windows clone came out in 2006, and 10 years has brought a lot of improvements, but ReactOS seems to us to still be stuck in no-man's land. It's more advanced than most hobbyist OSes, but it's not yet advanced enough to be useful as a main operating system just yet. In order to make the jump, it needs to offer something tangible that Windows doesn't.

This may seem like a tough ask for something that's a clone, but there are two areas where ReactOS potentially could develop into a serious offering: simplifying cloud licensing and resurrecting dead software. In the cloud, ReactOS seems to be struggling to gain interest. In 2014, the Thorium project tried to raise \$120,000 to fund development of a cloud desktop based on ReactOS. They

made it to an impressive \$48,000, but since then the project seems to have died. It's a shame, because confusion over licensing is holding Windows back in the cloud, so an open source project that could eradicate this problem while still being accessible to people who know the Windows environment could be very popular.

ReactOS offers better support for old software than modern versions of Windows. This has improved in version 0.4 with the *NT Virtual DOS Machine (NTVDM)* which enables old 16-bit applications to run on modern processors. While this is likely to be most useful to retro

ReactOS offers better support for old software than modern versions of Windows



Relive the glory days of *Minesweeper* with ReactOS.

gamers, the general support for older Windows software could also find mainstream use. It could be easier to keep a program written for Windows XP running on ReactOS than port it to Windows 10.

PROJECT WEBSITE
<https://www.reactos.org>

Thought organiser

Cherry Tree

Oh to have a brain like a computer. Wouldn't it be wonderful be able to allocate memory and have it remain in one place, always retrievable? You could create an array of dates called birthdays, and just append new elements every time you met someone new.

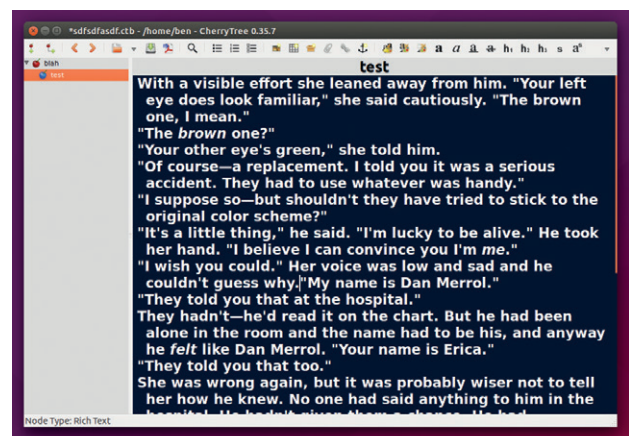
Perhaps it would be easier to set your head up like a database and create a table of people. This table could have columns for birthdays, favourite food and other key facts. Properly referenced keys could then refer to partners, children and other relationships... actually, we'd have to be a little careful with normalisation – it would probably be better to have a tables for families and households and reference those.

Alas, it's useless designing the perfect Boyce-Codd normal form

schema, because the simple truth is that our wet, soggy brains can't handle this level of data processing.

Since the mushy data inside our heads isn't formatted in anything approaching a normalised schema, if we want some way to store our thoughts digitally, it needs to be able to adapt the chaos in our minds. *Cherry Tree* is a hierarchical note-taking application that tries to apply just enough order to allow our thoughts to be transferred to a digital medium and be usable. You can create a tree structure of text notes, which nicely fits the flow of human thought. One branch leads to shopping lists, another goes to your projects, another holds plans to take over the world.

If your desk is under a sea of post-it notes and scribbled on bits of paper, *Cherry Tree* can help you



Unload your brain into your computer and never forget anything again.

restore some order and make sure your greatest ideas don't accidentally get recycled at the bottom of a pile of old notes.

PROJECT WEBSITE
www.giuspen.com/cherrytree

Instant messenger

Kadu

The world of instant messaging may now be dominated by huge multinational companies that produce closed source software with closed protocols designed to lock users into their platforms, but we're not yet ready to give up on the idea of free and open communications. *Kadu* is an XMPP/Jabber client that enables you to connect to any chat server that supports these protocols. Once upon a time that included Facebook chat, but alas, that is no longer possible thanks to the social network dropping support for chat software other than its own.

If you don't already have a chat account that supports an open protocol, you can find a wide range of public XMPP servers at <https://xmpp.net/directory.php> (with an account at one server, you

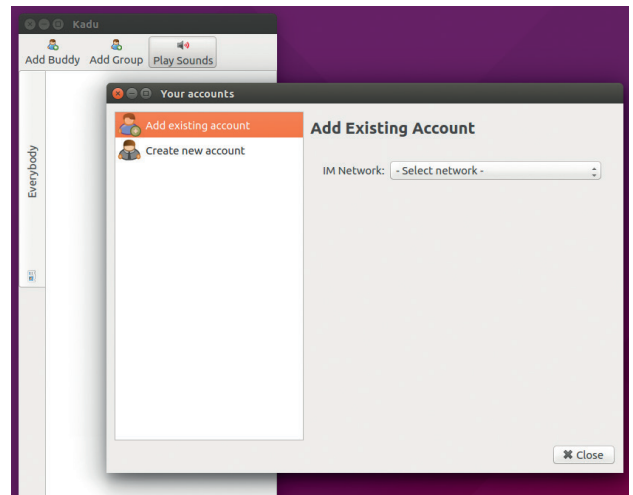
can chat with people on any other server).

Kadu supports all the usual features you'd expect from an instant messenger, such as the ability to manage multiple accounts, buddy lists and file transfers, and some more advanced features such as Off The Record (OTR) encryption for added security. Perhaps the biggest feature of *Kadu* is support for the Gadu-Gadu chat protocol, which is particularly popular in Poland.

Jak się masz

The *Qt 5* interface looks good, and works well. *Kadu* is most attractive

Kadu is most attractive to people looking for a Qt chat client that doesn't need KDE



French readers with poor spelling can enjoy this instant messenger client.

to people looking for a *Qt* chat client that doesn't require the KDE framework to be installed. The small set of dependencies make it a great option for lightweight desktops.

PROJECT WEBSITE
www.kadu.im

Download manager

uGet

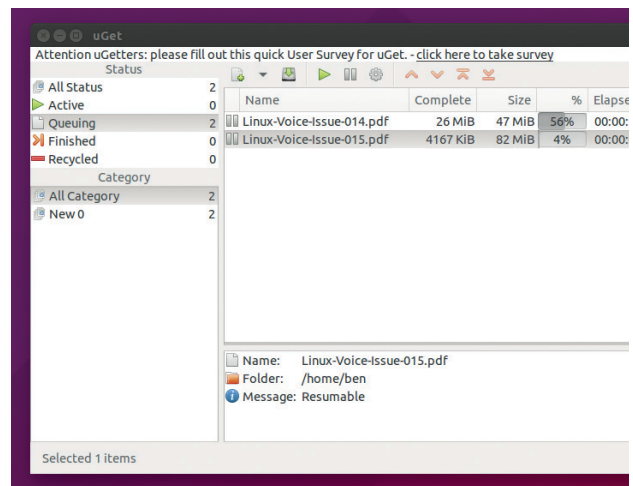
It seems to be a truth universally acknowledged that our internet connection is always slightly slower than we want it to be. Back in the mid 90s, 28.8Kbps was alright, but we would have loved an extra 10Kbps. Then, when ADSL came out, we always wanted another 1Mbps. In 2016, we're eyeing up BT fibre internet, but no doubt, next year we'll want something even faster.

Given that the internet is always, and will always be, too slow, we need software to help us make the most of our connection. As a download manager, *uGet* gives you lots of control over how you get large files over the internet. It won't automatically speed up your internet, but it will help you use your connection more intelligently. For example, imagine someone else in

your house is streaming a video when you suddenly come across a new distro that you want to download. You could just download the video and not worry about their video buffering, or you could try to remember to download the distro later. With *uGet*, you can add it to your download queue, and set it to start downloading in the middle of the night. You don't even need to worry about wasting electricity, because you can set *uGet* to shut down your machine when it's finished doing its work.

Download managers like *uGet* are for voracious data hogs like us.

The internet is too slow... we need software to help make the most of our connection



Mastering your downloads (and your anarchic downloads folder) is key to getting the most out of your internet connection.

We'll never be satisfied with our internet connection because the faster we can surf, the faster we can find more things we want to download.

PROJECT WEBSITE
<http://ugetdm.com>

Terminal image viewer

Termpix

If you think really hard, it's possible to come up with a use for an image viewer that runs in the terminal. Perhaps you might need one if you're ever SSHed into a server and need to check what pictures are available without downloading them. However, if this situation were to arise, installing a new bit of software would probably be more hassle than just SCPing the pictures to your local machine – especially if you have to install the dependencies (*Rust*) on the server.

Or maybe you're the developer of an operating system written entirely in assembly language that has no graphics system and can only handle text, and you need a method of converting images so they can be displayed in this environment.

Whether or not *Termpix* has any real use is a little irrelevant to us. It caught our eye because what it

does is interesting even if it's completely pointless. Essentially, the program is really a converter between images (of any format supported by the *Rust* image library), and strings of ANSI text containing the characters for half-block colours. This test is then printed on the screen to reveal a low-resolution version of the original image.

The end result is something akin to the charming graphics from 8-bit computers. Maybe that's a use for it after all: creating graphics for retro games. Be on the lookout for a 80s-style adventure game about a renegade group of Linux journalists



Coming next issue [maybe]: Linux Voice in glorious ANSI-text-rendered technicolour!

fighting against the forces of proprietary software. It'll be called *Linux Vice*.

Termpix caught our eye because what it does completely pointless

PROJECT WEBSITE
<https://github.com/hopey-dishwasher/termpix>

Screen reader

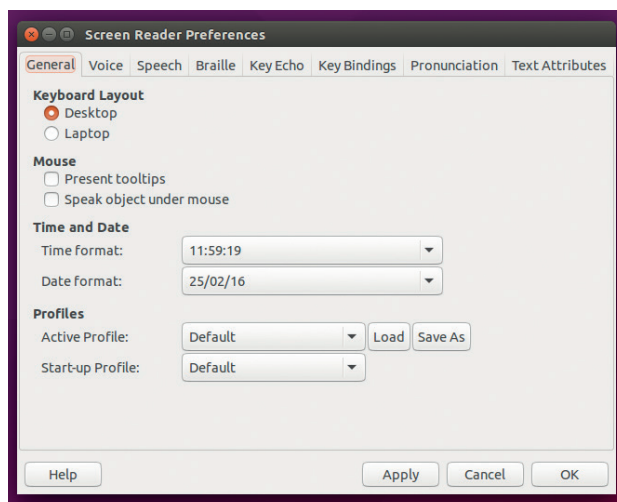
Orca

For most people, interacting with a computer means looking at a screen, tapping away at a keyboard and wiggling a mouse. However, humans don't all come from the same mould, and not everyone has the same set of senses. The *Orca* screen reader converts the text on screen to spoken words so that people who struggle to read a screen can still get the information off it.

Orca sounds fairly robotic, but the aim isn't to produce perfectly human voice, but to make computers usable for people with poor sight. *Orca* works well, but ultimately, not many applications are usable for someone without any sight, because most user interfaces aren't designed with visually impaired users in mind. Things like the Tab order, keyboard control and

having text as well as icons make a big difference with *Orca*.

As it is, *Orca* works well for people with some vision who need a little help. It could also be useful for fully blind people if used with carefully selected software that's been set up with *Orca* in mind. Perhaps the most useful application for *Orca*, though, is for developers. Remember that not all of your users have the same physical attributes as you, so while designing an interface, try using it with *Orca* and see how it fares. If you can't use your application with a screen reader, then what hope



You can change the default *Orca* setup by pressing **Insert + Keypad Insert + Space**.

does someone who doesn't know the software already have? A few simple changes can make a huge difference to some users – all it takes some thought.

The Orca screen reader converts the text on screen to spoken words

PROJECT WEBSITE
<https://help.gnome.org/users/orca/stable>

Display server

Wayland

2016 is slated to be the year that the *X windows* graphical display server gets replaced. Actually, 2014 was supposed to be the year *X* got replaced, then 2015 was, so we aren't holding our breath. So far, the only distribution to fully switch to a different display server is RebeccaBlackOS.

For those of you not familiar with the starlet after whom this distro is named, she hit a small amount of fame in 2011 with the song *Friday*. This was followed two years later by the unimaginatively titled *Saturday*. Fortunately, RebeccaBlackOS (RBOS) has almost nothing to do with Miss Black other than the default username of *beccaholic*.

As the only distro that's ditched *X* for *Wayland*, RBOS is far more noteworthy for its display server than its name. You can try out a

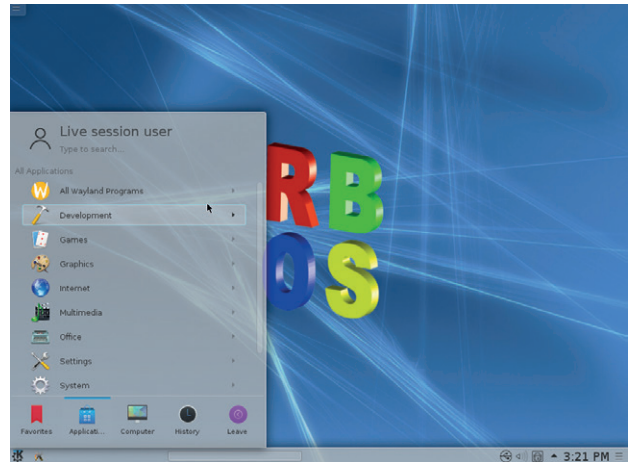
range of desktop shells on top of *Wayland* including Weston, KDE and one called Hawaii.

Crouching server

The difference between *Wayland* and *X* is almost entirely behind the scenes, and this is especially true when using a desktop that runs on both display servers, such as KDE.

In theory, the version of KDE running on *Wayland* should be identical to the version running on *X* except that it should perform better. This performance will be particularly noticeable in areas that are tricky for a display server, such as scrolling a web page with a

The difference between *Wayland* and *X* is almost entirely behind the scenes



We ran KDE on *Wayland* without problems, but heavy users may still find it unstable.

playing video on it. Ideally, you should see no screen tearing and lower memory and CPU load. We found *Wayland* performed well on our machine, but until it's used more widely, it's hard to know how it will perform in the wild.

PROJECT WEBSITE
<https://wayland.freedesktop.org>

Linux installer

Calamares

If you've been using Linux for a while, you probably don't give too much thought to the distro installers. Once you've installed your favourite distribution a few times, you can probably do it without even thinking. However, we should probably spare a thought for the first time Linux users as it can be a stressful time when you first put a new OS on your computer.

There are a lot of options, and they're not always well laid out. Part of this problem comes from the fact that most major distros build their own installers, despite the fact that they all do the same basic job. Each installer works slightly differently, and the work on each is duplicated.

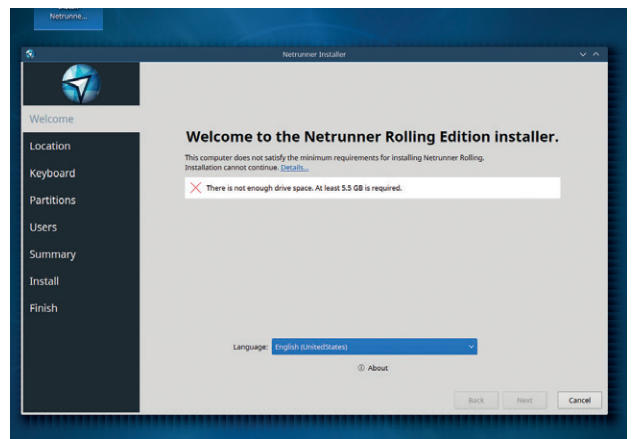
Calamares is a project to build a distro-independent installer. This doesn't mean that you can use it on

any distribution, but that any distribution can adapt it for their own needs rather than building their own from scratch.

By sharing the same installer, distributions can also share the effort needed to create and maintain the installer, which should mean a higher-quality installer for everyone while at the same time, leaving distro developers with more time to focus on what makes their distro great rather than messing around with an installer.

Quite a few Linux distributions are already using *Calamares*, including NetRunner, Manjaro and

Most major distros build their own installers, despite them doing the same job



The popular KDE distribution NetRunner uses *Calamares* rather than building its own installer from scratch.

Sabayon. Even Fedora is currently evaluating its usefulness for future KDE remix versions. Keep an eye out and you might come across *Calamares* on a distro near you in the near future.

PROJECT WEBSITE
<https://calamares.io>

FOSSPICKS Brain relaxers

Real-time action game

OpenClonk

Clonks are members of a humanoid race who seem to be in need of a little help. You can guide your clonks through a 2D world on various missions as they build, mine and fight. Alternatively, you can play against other users to see who is the mightiest Clonk controller.

Gameplay is a bit like a more involved version of the arcade classic *Worms*. Your Clonks can collect different bits of equipment, and you have to direct them to use it. The missions, though, are more complex and require more work to complete. Don't worry if this all sounds a bit confusing, there's a full set of tutorials to help you get started.

OpenClonk is a direct continuation of the *Clonk* series

of commercial games that began in 1994. This series ran for an impressive 20 years and nine different versions. These closed source games are still available as freeware (not open source) from www.clonk.de. After the lead developer left the project, the company behind the game (Redshift) decided to discontinue the series. However, it released the source code for the game engine under an open source licence, so the spirit lives on in *OpenClonk*.

As well as being a game in its own right, *OpenClonk* is an interpreter for the C4Script

You can do everything from building your own levels to creating entirely new games



Dig, mine and fight your way to primitive humanoid supremacy as the ultimate *Clonk* leader.

language. The enables you to do everything from building your own levels to creating entirely new games. If you want to get stuck into creating your own Clonk world, you can find everything you need at http://wiki.openclonk.org/w/C4Script_Documentation.

PROJECT WEBSITE
www.openclonk.org

Real-time strategy game

Megaglest

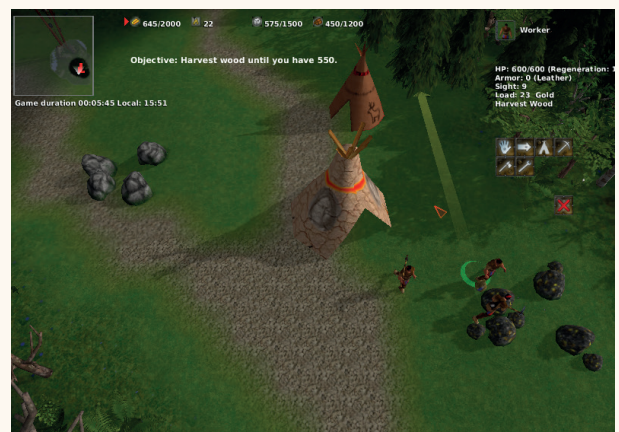
Megaglest is a 3D real-time strategy game in which you build and train your army, then enter battle against one or more enemies. As the name suggests, *Megaglest* is built upon *Glest*, an earlier open source game that is no longer developed.

Although there have been improvements across the board, from the player's perspective, the biggest improvement in *Megaglest* over *Glest* is the increased amount of gameplay. As well as the official game which includes six different factions to play as (Tech, Magic, Egypt, Indians, Norsemen, Persian or Romans), there are a wide range of user-contributed expansions including Japanese,


British, Crusaders – and of course, what open source game would be complete without the ability to play as Penguins?

The graphics look a little dated, but we don't believe that game graphics correlate in any way with the enjoyability of computer games. It doesn't matter if you can ray-trace in real time, if the gameplay isn't fun then the game isn't fun and conversely, horrendous graphics with great gameplay still make an enjoyable game. We're pretty sure that gaming reached a peak with 8-bit CPUs, and if anything, *Megaglest* is a little too realistic. Perhaps we could route the graphical output through *Termpix* for a retro experience.

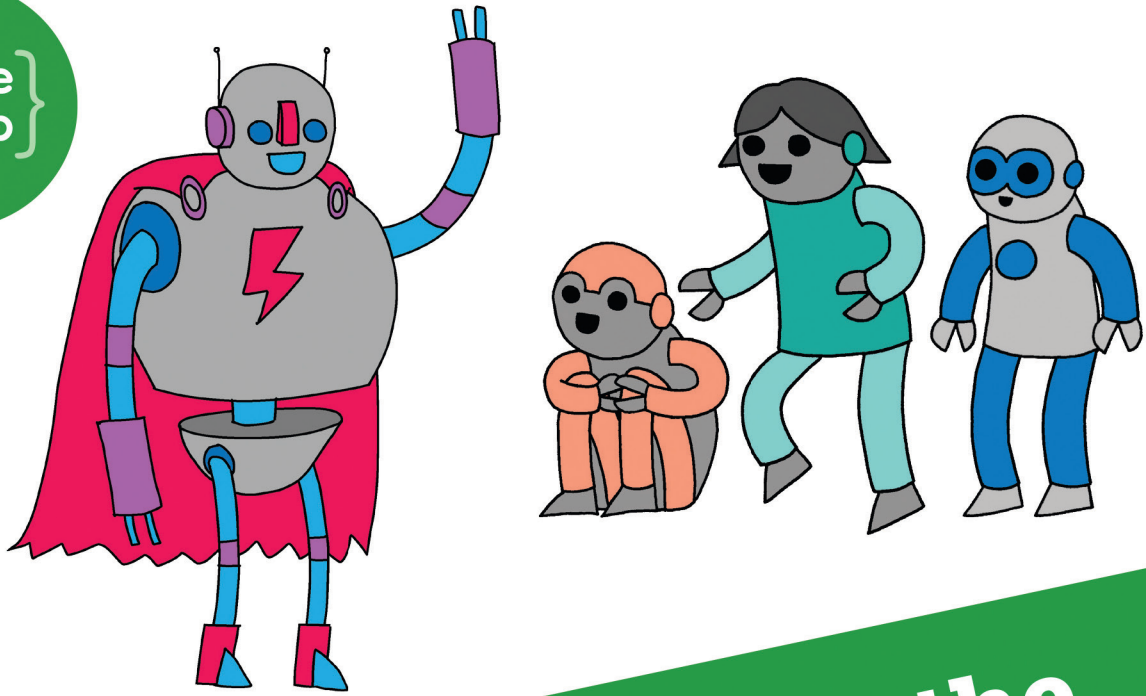
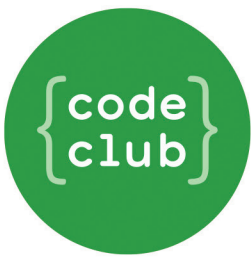
If you'd rather spend your time being sociable than working out



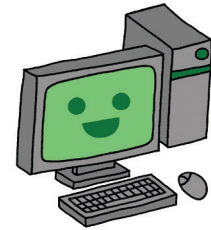
The range of different factions and scenarios means that there's plenty to keep even the most ardent engaged in *Megaglest* for some time.

elaborate ways to downgrade the graphics to regain the lost feelings of youth, there's an online community of players, and there's usually a game on somewhere for you to join. 

PROJECT WEBSITE
<https://megaglest.org>



Can you help inspire the next generation of coders?



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

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

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

There are loads of ways to get involved!

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

TUTORIALS

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



Ben Everard
Can be beaten by the Go AI on a particularly sluggish Commodore 64

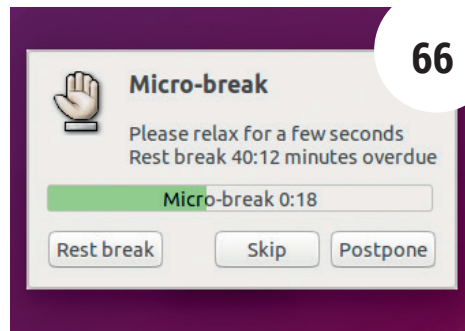
One of the big milestones in artificial intelligence fell this month. A computer beat a world champion in Go, a board game with simple rules but very complex tactics. While this is an impressive feat of computing, it's always felt a little hollow to me to attach the tag intelligence to this sort of computing. Intelligence isn't the ability to be taught how to strategise by analysing millions of scenarios in a contrived environment, it's the ability to thrive in new situations and apply novel approaches to solve old problems.

AlphaGo (the computer that won the match) could only be considered to be intelligent if it could take the lessons learned from Go and apply them in other areas: chess perhaps, or business strategy. Reprogramming the machine to work in another area isn't sufficient; it has to be able to adapt by itself and transfer the skills of one area to another just as humans do every day.

If computing's going to really move forward, we need to stop focussing all our efforts on artificial intelligence and start trying to develop actual intelligence.

ben@linuxvoice.com

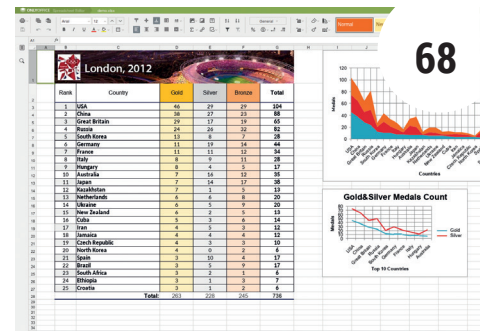
In this issue . . .



66

Schedule your work to keep RSI at bay

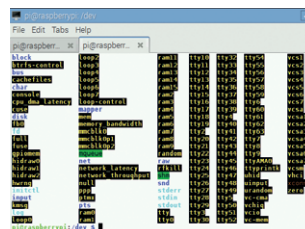
Ben Everard sometimes has sore hands, so he takes regular breaks from work to ease the strain. *Workrave* makes sure that he doesn't forget.



68

Put your office suite in the cloud, but stay private

You don't have to hand over your data when you put your office suite online. **Valentine Sinitsyn** uses *OnlyOffice* to build his own cloud.



72

Terminal velocity

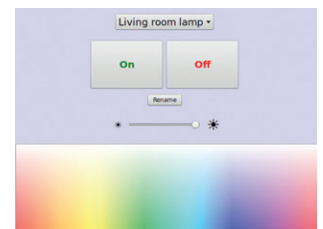
Les Pounder introduces the awesome power of the Linux command line interface.



76

OpenStack

Not content with one computer, **Graham Morrision** runs 10 on a single CPU.

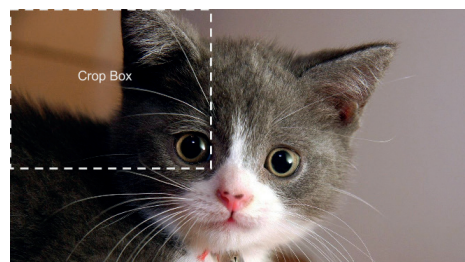


80

Let there be light

The ever-illuminating **Mark Crutch** finishes his series on light control with Linux.

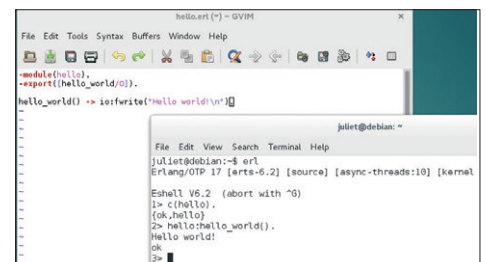
Coding



84

Ready, steady ... Go!

Trendsetter **Amit Saha** discovers a new language from Google and likes what he finds.



90

Languages of the future

Juliet Kemp enters her virtual tardis and sets forth to find the programming languages of tomorrow.

EASE FINGER STRAIN WITH WORKRAVE

Stay healthy and pain-free by taking breaks from work.

BEN EVERARD

WHY DO THIS?

- Keep eye- and wrist strain at bay
- Go outside, smell the flowers, feel the rain and the sun on your vitamin D-deprived face.
- Bear the lead-heavy haunches of the toad and work more lightly

Keyboards, mice and monitors are the link between people and computers – the meat-space equivalent of a USB cable. Just like a USB cable, this interface can wear out if it's used too much or not looked after properly. We may be able to get new parts for our computers when they wear out, but getting new hands and eyes if our bodies start to wear out is much more difficult. It's important, then, to take the best possible care of our meaty interfaces so that we can keep communicating with computers

for as many years as possible (or at least until we get a more robust human-computer interface such as a USB port directly in our cranium).

There are many aspects to taking proper care of our body, and in this tutorial we're going to look at one: timing. All communications protocols have optimal timings, and the human-computer interface is no different. The best option isn't just to blast the information across as quickly as possible, but to work in a way that minimises the strain on our joints.

STEP BY STEP: LOOK AFTER YOURSELF

1 Install Workrave

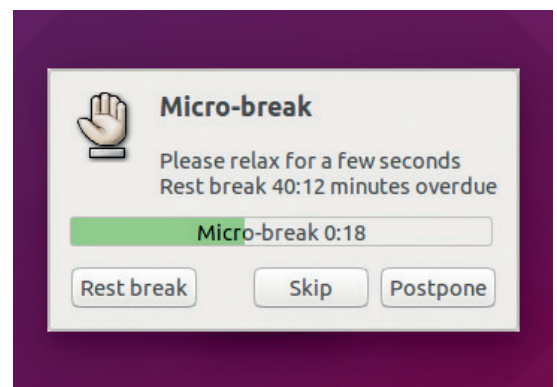
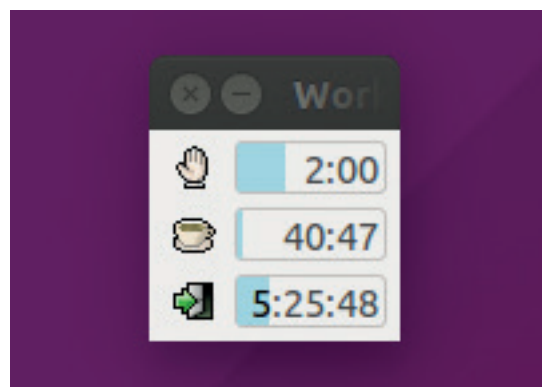
Workrave is an activity monitor that's designed to interrupt your computer use to remind you to take frequent breaks – these breaks are important for the health of your hands and eyes. You might find it in your distro's package manager, but if you don't you can download a tarball from the project's website (www.workrave.org). Installation is via the standard method of unpacking the tarball then running `./configure, make` and finally `sudo make install` in the unpacked directory.

If you're unfortunate enough to have to run Windows, you'll also find an install file on the project website. There's not currently a version of *Workrave* for Apple's OS X, though there are some similar applications such as *20 Cubed* (<https://chrome.google.com/webstore/detail/20-cubed/geghmabifcdlkmnpnkapfefbbfaonhcef>) and *Awareness* (<http://iamfutureproof.com/tools/awareness>).

2 Take a Microbreak

Microbreaks are the smallest counter on *Workrave*. By default, every three minutes, *Workrave* will prompt you to take a 30-second rest. First, you'll see a popup prompting you to take a break, and this will then monitor your keyboard use. If you stop using the keyboard, *Workrave* will start a 30-second countdown, during which you won't be able to access anything on your computer. The three minutes between microbreaks only counts time you're actually using your computer. If you stop using the mouse and keyboard, then the countdown will pause.

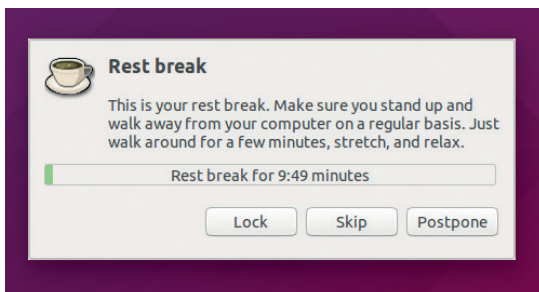
The key aspect of these short breaks is to briefly rest the parts of you interacting with the computer, that is, your hands and eyes. While your hands will automatically rest when not using the keyboard, your eyes will only rest if you look away from your screen. Ideally, you should find something much further away from you than your screen to give your eyes a break.



3 Take a rest break

Rest breaks are 10-minute breaks that come about every 45 minutes. During these, you should take the opportunity to stand up and get away from your computer. By doing this, you'll stimulate the muscles in your back and arms in ways that will relieve the tensions that can build up while sitting at a desk. We like to use some of our rest breaks to do finger exercises that help to strengthen our hands and minimise the damage caused by typing.

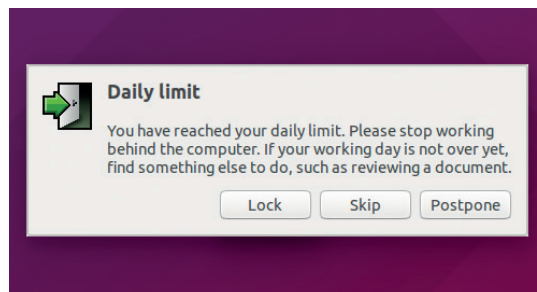
You don't have to take your breaks exactly when *Workrave* suggests: sometimes, you might need just a few more minutes to finish working on something while you still have the train of thought in your head. For this, there's a Postpone button that enables you to delay the break in the same way a snooze button delays an alarm clock. You can't just delay indefinitely though, because you're only allowed three postpones per break before it becomes mandatory.



4 Finish for the day

No matter how many breaks you take, there's a limit to how much computer use you do in a day before you'll start run the risk of serious problems. *Workrave* recommends four hours per day of actual computer use (typing or mouse control). Again, this only counts active computer use rather than all the time that your computer is switched on. You can postpone this three times if you're just coming to the end of a critical piece of work.

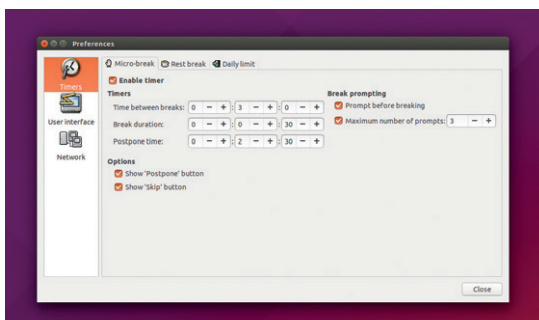
Finishing for the day means finishing computer use. Depending on your job, there may be several things that you can do without using a computer. For example, we like to proofread printouts, plan projects on pen and paper and take care of anything that needs to happen in the real world. By limiting the amount of time you spend at the computer each day, you can dramatically reduce the amount of strain on your body.



5 Configure Workrave


In truth, we're only just starting to understand the effect that long-term computer work has on the body. While most experts agree that taking rest breaks is essential to ensuring your long-term health as a heavy computer user, there's no clear consensus on the best way to schedule your breaks. *Workrave* doesn't mandate the periods of each break, and allows you to configure most how long to rest for and how frequently to rest. You can enter the preferences window by right-clicking on the *Workrave* window and selecting preferences. If you're already suffering from RSI or some other strain, you may need to break longer or more frequently. Alternatively, you might feel it more appropriate for your work to break less often.

If you use multiple computers in the course of your work, you can also configure *Workrave* in network mode so that it takes into consideration your work on all your machines.



6 Other options

Workrave isn't a complete solution to staying healthy while using a computer, it's just one tool to help in one area. To ensure you have many years of productive computer use a head, you need to ensure that you properly take every possible precaution against computer-related injuries. The most basic is proper desk setup where you ensure that you have your seat, desk and monitor all at the appropriate height. Further precautions include using ergonomic keyboards and mice, and considering alternative desk setups such as standing desks. There isn't a standard setup that's right for everyone, so you'll have to do some research to find out what works for you.

Above all, the most important part of staying safe when using a computer is not ignoring any warning signs. If you get any tingling, numbness or pain when working, you should immediately look to mitigate the problem and ideally seek medical advice. 



ONLYOFFICE: HOST AN OFFICE IN THE CLOUD

Get all the power of online office suites without having to trust a third party.

VALENTINE SINITSYN

WHY DO THIS?

- Enhance privacy – your data is yours.
- Work from almost any device, anywhere.
- Explore free office suites beyond *LibreOffice*.

Marketing guys say we live in a "cloud first" era, and they are probably right. Many of us rely on web-based, remotely hosted apps for daily tasks, like reading email, writing texts or working with spreadsheets. Having all your data in a single place accessible from everywhere is good for productivity, but bad for privacy. And if you are into Free Software, you probably care about privacy.

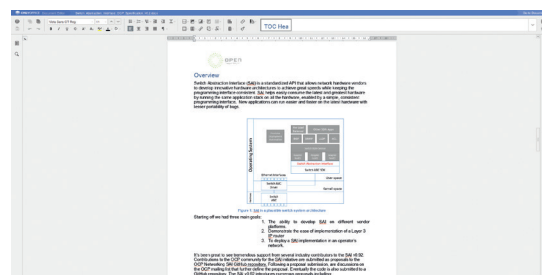
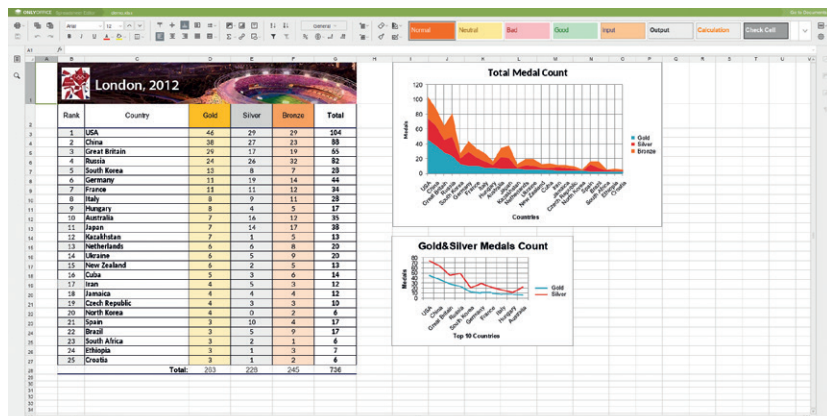
Is there a way to bring together the best of both worlds? The answer is yes – to a certain extent – and is often called self-hosting. You still use web applications, but run them on your own servers. Your files are under your control, and you only need a web browser to access them wherever you are.

OnlyOffice is a hosted alternative to online office suites like Google Docs, Microsoft Office Online or Zoho Documents. It's also available in the cloud, if you wish, and best of all, it's free (as in speech). We reviewed it in LV017, and in this tutorial, we'll show you what is it capable of, and how to deploy it on your network with minimal efforts.

Getting your feet wet

OnlyOffice started in 2009 as *Teamlab*, a proprietary Windows-based product. Five years later it was renamed, released under AGPLv3, and brought to Linux, thanks to Mono. Today, *OnlyOffice* comes in three main flavours: Free, Enterprise and SaaS. We are going to cover the first; Enterprise is a commercially hosted solution, and SaaS is a cloud version (both free and paid-for plans are available). They aren't identical, as new features often appear in the Enterprise or

The *OnlyOffice Spreadsheet Editor* showing a sample document with London 2012 Summer Olympics stats.



The *OnlyOffice Document Editor* displaying SAI specification from Microsoft *et al*. Despite being free (Apache v2), it comes in OO XML rather than PDF format.

SaaS first. The project's website is at <https://www.onlyoffice.org>.

OnlyOffice is actually an umbrella term for several interconnected products. At its core is the *Document Server*, which is a family of online document editors. *Document Server* provides a word processor, spreadsheets and presentations, and that's it. To manage your documents, you'll need *Community Server*, which also brings calendaring, CRM and project management features. Put simply, *Document Server* is your traditional desktop office suite brought to a browser, *Community Server* is a collaboration/groupware tool built on top of it. Communications are supported via the *Mail Server* add-on, and *OnlyOffice Talk*, an XMPP/Jabber service built into the *Community Server* itself.

There are several ways to deploy *OnlyOffice*; we'd use *Docker*. Note that *OnlyOffice* has known incompatibilities with older Linux kernels. If you run an LTS distribution, it's better to upgrade to the latest kernel first. A system with dual-core 2GHz CPU and 4GB RAM is recommended to run *OnlyOffice* in production. For evaluation and testing, half of these specs should probably suffice.

Ensure you have *Docker 1.4.1* or later installed. If not, either use your package manager or get it directly from www.docker.com. *Document Server* official images are on Docker Hub, so you can get it running with a single command:

```
$ sudo docker run -i -t -d -p 80:80 onlyoffice/documentserver
```

Depending on how your host is configured, you may not need the initial **sudo**. *Document Server* will

listen on port 80; change the first number in **-p** if you want anything else. Now, open your web browser and navigate to **http://your-host-name-or-IP-address**. You could use localhost or 127.0.0.1 for now, but *Community Server* gets confused with them. So, better use a real (resolvable) hostname or the host's IP address, like 192.168.1.x.

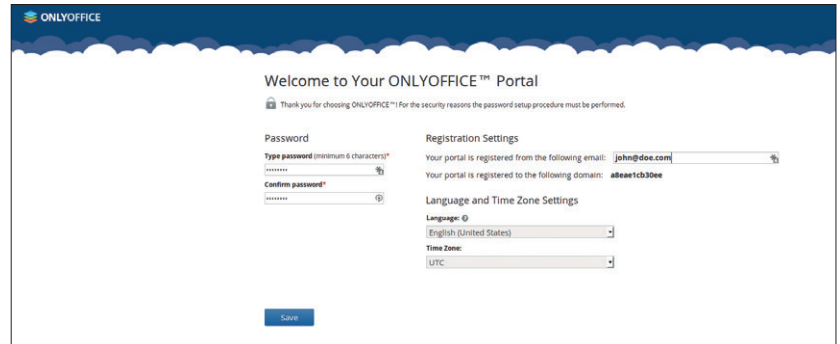
Meet the editors

By itself, the *Document Server* is really bare bones. You can upload a file from a local PC, edit it and download back – that's it. You can't create new documents, delete them, or manage files in any way. There is also no access control. To make long story short, *Document Server* is more of a technology demo than an online office suite. You'd want *Community Server* for any real deployment.

The native document format for *OnlyOffice* editors is OO XML. This matches *OnlyOffice* to Microsoft's offerings but makes Richard Stallman sad. You can upload documents in almost any format, including OpenDocument, or legacy DOC, XLS and PPT. Behind the scenes, *OnlyOffice* calls *LibreOffice* to convert anything you throw at it to OO XML. It is also possible to download your documents as OpenDocument files. *OnlyOffice* will even try to open PDFs for you, although it never succeeded in our quick tests. Yet it handled a complex DOCX file that *LibreOffice* failed to render properly – your mileage may vary.

OnlyOffice has most of the features you'd expect from an online office suite. You have all the usual text formatting options on the first three ribbon palettes. It is possible to insert images (either from a file or URL), tables and charts and create autosshapes and hyperlinks. A useful option is "Insert text", which creates a text frame, as in desktop publishing software. There is also a quite sophisticated equations editor, and some styling tools. For instance, you can change the document's colour scheme (which affects charts, autosshapes and tables), or create a heading with one mouse click.

Sometimes, *OnlyOffice* feels rather counter-intuitive. To create a new style, you format some text the way you want, then save it as a style, not the other way around. And it doesn't work in *OnlyOffice Free*,



at least for now. Comments and changes tracking are also not in the Free edition yet, which is a pity. The equation editor may feel inconvenient, if you're used to *LibreOffice Math* or *Latex*. Finally, automatic tables of contents aren't supported. This means that *OnlyOffice* is hardly suitable for editing large documents, like books or reports.

Two other *OnlyOffice* components hide no surprises. Even the ribbon stays almost the same. Formulas in spreadsheets work, and various slide layouts and themes are available for your presentations. *OnlyOffice Free* doesn't seem to support macros, which is understandable, yet may disappoint some business

Before you can use the **Community Portal**, some things need to be tweaked. Make sure to enter a real email address.

At the core of OnlyOffice is the Document Server, which is a family of online document editors

users. It also turns out that the spreadsheet editor doesn't play well with *Firefox's* "Find as you type" feature. When it is on, the search box appears at every keystroke, seizing the focus. You'd want to disable this feature during your *OnlyOffice* sessions.

Create a portal

Community Server builds on the *Document Server* to add portal features like document management, access control, and third-party file hosting support. It also comes with extra tools, including a calendar, a project management solution, a CRM and *OnlyOffice Talk*, albeit we won't cover them in this tutorial.

Your office, your way

It is possible to embed *OnlyOffice* editors in your own software. So, if you like the editors but dislike the *Community Server*, there's nothing to stop you from making your own one that works exactly the way you want. That's what we like about Free Software.

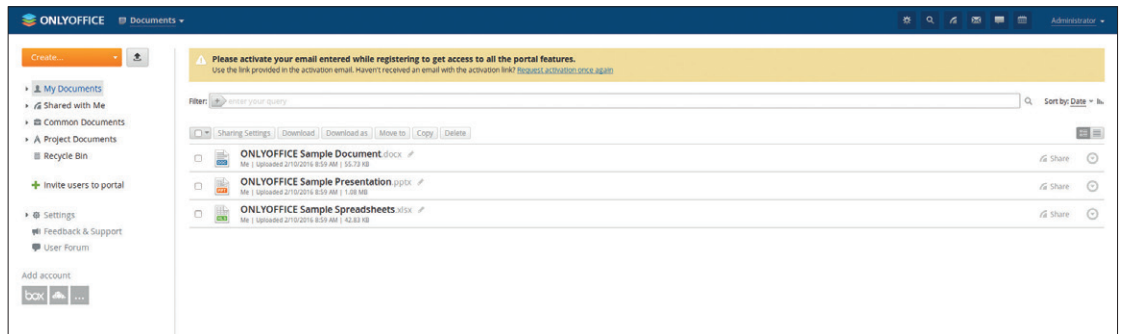
To embed the editors, you must have the *Document Server* up and running. The client builds on JavaScript (naturally) and the HTML 5 Canvas element. The later is somewhat close to how desktop apps work, and sets *OnlyOffice* editors apart from many cloud office suits.

To embed *OnlyOffice*, you include a JavaScript library and wrap a placeholder `<div>` element. Surprisingly, this is already

enough to view a document in any supported format. To save it, you should also provide some back-end, namely a callback handler that runs when the document is saved. The idea is that you implement a document storage service component, which makes documents available to *Document Server* (subject to permissions check), and also downloads them from the *Document Server* when anything changes.

OnlyOffice provides a rich API to customise the editors' look and feel, and also exposes an API for file format conversions (this really calls into *LibreOffice*, you know). You will find a detailed reference documentation along with usage examples at <http://api.onlyoffice.com>.

The *Documents* module is where you'll spend most of your time. You can make it a default start page for that reason



Official manuals suggest running *Community Server* in a separate *Docker* container linked to *Document Server* and *Mail Server* containers. We'll do it this way, albeit without the *Mail Server*. In a nutshell, *Mail Server* is just *iRedMail* plus *Postfix*, *Dovecot*, *SpamAssassin* and *ClamAV* wrapped as *OnlyOffice* add-on. It's really optional, as *Community Server* should integrate well with your existing email infrastructure.

Stop and remove the existing *Document Server* first, then recreate it with the explicit name and no ports exposed:

```
$ sudo docker run -i -t -d --name onlyoffice-document-server onlyoffice/documentserver
```

The official *Community Server* image at Docker Hub is outdated (8.5 vs 8.7), and it didn't work well for us. So, we recreated it locally from the latest packages:

```
$ git clone https://github.com/ONLYOFFICE/Docker-CommunityServer
```

```
$ cd Docker-CommunityServer
```

```
$ docker build -t communityserver:8.7
```

This takes some time. When the process finishes, run the *Community Server* container with:

```
$ sudo docker run -i -t -d -p 80:80 -p 5222:5222 -p 443:443 --name onlyoffice-community-server --link onlyoffice-document-server:document_server -v /opt/onlyoffice/Data/certs:/var/www/onlyoffice/Data/certs communityserver:8.7
```

Here, we expose ports 80/443 for HTTP/HTTPS access, and port 5222 for *OnlyOffice Talk* (XMPP). Mount volumes (-v) are needed to set up HTTPS (see below).

If you agree to stick to official images, the process could be much simpler. First, you may use *Docker Compose* to run all three containers (*Document Server*, *Community Server* and *Mail Server*) with one command. Second, you don't have to type any commands at all: you can use the *OnlyOffice* One Click Install (<https://one-click-install.onlyoffice.com>) automated service. We haven't tried this path ourselves; if you do, we strongly suggest you change the password or SSH key you posted to One Click as soon as the installation completes.

In a similar vein, we don't recommend that you access *OnlyOffice* via HTTP in production. Setting up HTTPS is simple; see the *Community Server's* README for details. Note that you should mount the **Data/certs** volume, not plain **Data** as the README suggests. Otherwise, *Community Server* won't be able to find its static files (scripts and stylesheets).

Make yourself at home

Now, open <https://your-host-name-or-IP> (remember not to use localhost!). *Community Server* takes some time to initialise, so if this page doesn't open, wait a second and try again.

Community Server needs some initial setup. On your first visit, it will ask you for the administrator's email and password, the language and the timezone. When you'll get an email from *OnlyOffice*, follow the link to activate the Administrator's account. For testing, you may skip this step, but you won't be able to get portal notifications or change your credentials then.

OnlyOffice isn't the only...

At the time of writing, *OnlyOffice* was arguably the most mature free online office suite. But it's certainly not alone.

The first attempts to run *LibreOffice* in a browser date back to 2011. The original prototype used a *GTK 3 + HTML 5* (Broadway) back-end. This approach didn't scale well, and was tying clients to the server too much.

So, in 2015 Collabora teamed up with IceWarp to build the new *LibreOffice Online* (LOOL). For starters, *Writer*, *Calc* and *Impress* were included, but this list may expand in future. The project reuses as much from the existing *LibreOffice* codebase as possible, and renders your documents for the web the same way it does for the desktop. The *LibreOffice Online* back-end calls into a thing called *LibreOffice Kit*, which draws document tiles, much like map services do. These tiles are sent to the front-end communicating via a *WebSocket* connection. This means better scalability and less coupling between the client

and the server. Recently, an *OwnCloud* plugin for *LibreOffice Online* was announced.

The Document Foundation also seems to have no plans to build a cloud service on *LibreOffice Online*, so it is likely a "hosted-first", not "cloud-first" solution. Yet you can try it yourself, as Collabora provides CODE (Collabora Online Development Edition) – an OpenSUSE-based VM image containing the latest developments in the field. It also includes *OwnCloud*, so you can test both pieces of the puzzle together. Download links and detailed instructions are on <https://www.collaboraoffice.com/code>.

CODE is bleeding-edge, and not everything may work properly. We got occasional socket errors, and were not able to run the VM on IPv6 network. Yet we hope CODE will help you to build a picture of the LOOL's state today. Give it a try, file bug reports, send patches and stay tuned for the updates!

When you log in to the portal, it will show you the module choice page: go to Documents. To make *Community Server* start from this page, click on the gear icon, choose Common > Modules & Tools, and set Documents as the default start page.

To create a new document or folder, use the menu just below the *OnlyOffice* logo. A button next to it uploads documents from a local hard drive. Remember that *OnlyOffice* converts documents you upload to OO XML, although you may choose to keep the original as well. My Documents is where you find your own files, and *Common Documents* stores files for all portal users. You can also share your private files with portal and non-portal users, and control access at user or group level. The only file management feature we miss is a Recent Files list.

To open a document, click on its name in the list. An editor window will appear, and you can start making changes. Documents are saved automatically. When you are done with the document, just close the tab or click the "Go to Documents" link in the top-right corner.

Note that you're working as Administrator now. To add some unprivileged users, choose People from the drop-down near the logo. From the People module, you can add users or groups, and import them from your cloud account or CSV file. You can also simply send email invitations to those you want to collaborate with.

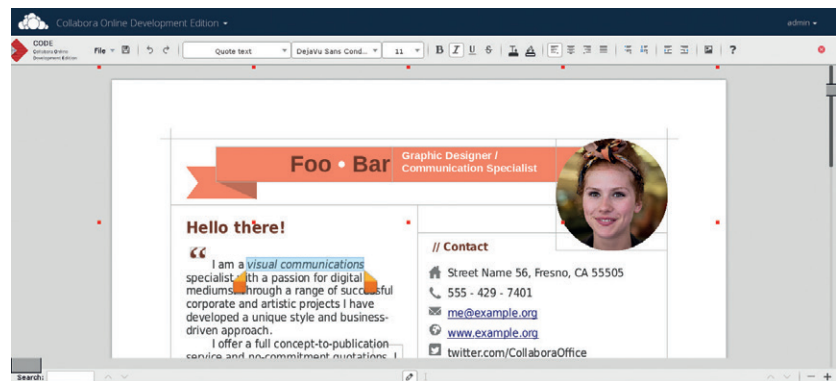
Add some storage

If you already use some online office suite, you'd probably want *OnlyOffice* to access your existing documents in the cloud. *Community Server* supports integration with third-party file-hosting services like Google Drive, Dropbox, OneDrive, and Box.com (among others). You may have already spotted their icons at the bottom-left of the portal.

By default, *CommunityServer* can set up WebDAV-based integrations only, as everything else needs API keys. So, navigate to Settings > Integration > Third-party Authorisation. You'll see numerous fields for authorisation keys related to different services. To get the required credentials, you'll need to register a developer's account with the service of your choice. This is usually free yet is a somewhat tricky process. Luckily, there are step-by-step guides available at <http://helpcenter.onlyoffice.com> (search for "Authorisation keys").

When you fill authorisation keys for some service, a respective icon will appear at the bottom-left of the portal. Click on it, and follow the on-screen prompts. Remote services look like folders at the Community Portal, and you have an option to make them common for all (or some) portal users. To change this setting later, you'd have to delete the folder (your data is not affected) and re-create it from scratch, so think carefully.

Having a hosted office suite that stores documents in the cloud doesn't help your privacy much. You'd also want a hosted document storage. A natural



choice here is *OwnCloud* (www.owncloud.org): it's popular and free.

We assume you already have *OwnCloud* running on your network. If not, there are numerous manuals available, not to mention an official *Docker* image. *OwnCloud* uses WebDAV for integrations, so you don't need any API key. Just click on the icon and fill in your *OwnCloud* login and password. The connection URL is found under the Settings link in the *OwnCloud*

LibreOffice Online brings the office suite we all know and love to the web browser (*OwnCloud* app included).

If you already use an online office suite, you probably want OnlyOffice to access your existing documents in the cloud

web interface. The name you use to connect to *OwnCloud* should be in *OwnCloud's* trusted domain list. Try to open it in your browser, and *OwnCloud* will guide you through the process. Remember not to use localhost, unless *Community Server* and *OwnCloud* run uncontainerised on the same physical machine.

Third-party services are essential for collaboration. This being said, *Community Server* is rather useless at conflict resolution. It may happily overwrite the file you changed outside the portal, or fail to upload your document to Dropbox for no visible reason. The latest *OnlyOffice* document editors (not available in *OnlyOffice Free* yet) bring real-time collaboration, so these issues should go away some day. Until them, it'd be better to keep them in mind.

A bit of summary

We hope this tutorial helped you to build an understanding of what *OnlyOffice* is. The technology looks promising, albeit with some glitches that may turn into blockers in specific use-cases. The free self-hosting edition is a real bonus, however it may feel more like a teaser for *OnlyOffice Enterprise* than a standalone product. We won't replace our good old *LibreOffice* with it just now, but we'll keep an eye on *OnlyOffice's* progress. If you already use an online office suite, *OnlyOffice Free* is a viable alternative. 

Dr Valentine Sinitsyn got addicted to office suites when he was the editor of Linux Format Russia. These days, he runs Vim or Python much more often.

LINUX FOR LEARNERS: TERMINAL BASICS

Les Pounder takes us on a journey into the Linux command line.

LES POUNDER

WHY DO THIS?

- Learn Linux.
- Update your Raspberry Pi.
- Navigate the filesystem.
- Create files and directories.
- Copy and move files.
- Rename and delete files.
- Add users to the system.
- Add users to super user group.

TOOLS REQUIRED

- A Raspberry Pi running the latest Raspbian release.
- An internet connection.

When we think of the Raspberry Pi we initially think of robots, coding and making cool projects. But what is underneath all of these projects? Why, it's the humble Raspbian operating system. Raspbian is based upon Debian, a solid Linux operating system that has powered everything from PDAs to huge server farms.

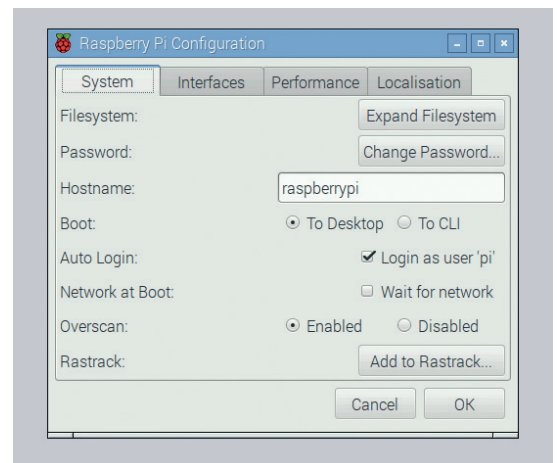
The easiest way to install Raspbian on a Raspberry Pi is to use NOOBS (New Out Of the Box Software). For this you will need to download NOOBS on to a PC. NOOBS comes as a Zip archive, which will need to be extracted to a FAT-formatted SD card with 8GB capacity. With NOOBS on your SD card, insert it into your Raspberry Pi and boot up. On first boot you'll be asked to choose an operating system. Tick the box next to Raspbian and then click Install. The installation process can take around 10 minutes, so now is great time to get a cup of tea. Once completed, the Pi will reboot and load the Raspbian operating system.

Expand the SD card

With the Raspberry Pi booted to the desktop we'll now perform a little maintenance. It's important that we give Raspbian as much disk space as possible. From the Raspbian desktop you can see the menu in the top-left of the screen. Left-click here and navigate to the Preferences menu. In the new sub menu you can see the Raspberry Pi Configuration application. Click on it. In the new window, you can see the Filesystem entry; left-click on Expand Filesystem To action. Once this is done, you will need to reboot for the actions to take effect.

On a typical Raspbian install there is only one user, called "pi". This user has a home directory at `/home/pi`. Your work, images, videos etc are all stored in this home directory. In this home you can create, edit, delete and move anything without needing any administrator powers. This is because it is full of your files. Each user has their own home and typically can only work on the files contained in that home. But there is one user who can do what they want, and that is the "superuser" sometimes known as "root".

The superuser can do anything: configure the system, delete any files and create users. But just like a Super Hero, with great power comes great responsibility, and when using this power you should



The new Raspberry Pi Configuration tool replaces the old *raspi-config* menu and provides an easy to use tool for configuring every aspect of your Pi.

be very careful! To temporarily give a normal user super powers we use a command called **sudo** before the command that we wish to run with that power, so for example let's open *LXTerminal* and run the **ifconfig** command with super powers.

\$ sudo ifconfig

Using **ifconfig** as an example is quite safe, as the basic command just lists the Wi-Fi and Ethernet connections for your Pi. We're not interested in the output; this command is just to illustrate how to use **sudo** with a command. We'll use **sudo** for real later in this tutorial.

Update your Pi

It's essential to keep your Raspberry Pi up to date. By updating you ensure that your Pi gets the latest software and important security updates. Only a user with super powers, **sudo**, can update the Raspberry Pi, as this is an administrative task that can affect the entire system. To update the list of software that your Raspberry Pi can install we need to check our list of software against a remote server. Open *LXTerminal* and type.

\$ sudo apt-get update

You will see lots of text whizzing along the screen; this is your Pi talking to a remote server and updating the software list on your Pi. Now this action does not

update the software, for that we need to issue another command, so in the same terminal, and after the previous command has completed, type the following.

sudo apt-get upgrade

After a few moments the screen will update and tell you what software requires upgrading. To confirm you are happy with this, press the Y key and then Enter. How long the action will take depends on your internet connection and how many updates are required.

A top tip for this task is that you can chain the two previous commands together. Once the first command, **apt-get update**, is completed it is tested to ensure that it has completed correctly. If that is the case then the second command, **apt-get upgrade**, is run. We then add **-y** to the upgrade command to automatically upgrade the new software. The command to do this looks like this.

\$ sudo apt-get update && sudo apt-get upgrade -y

It's important to keep your Pi up to date and this action should be done at least once per month.

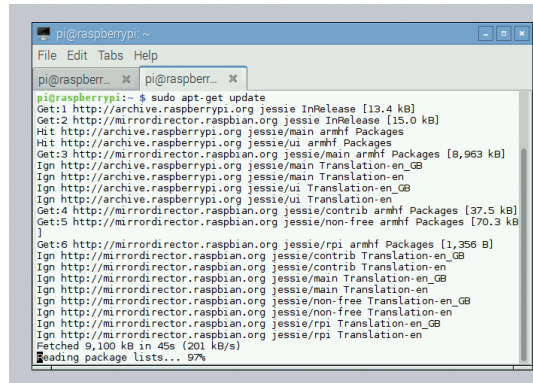
Basic Linux skills

There are many commands in Linux, some of which you'll use every day. So let's open up *LXTerminal* and try them out.

First, when we open *LXTerminal* we will see a interface that looks something like this.

pi@raspberrypi ~ \$

So what does that all mean? Well first we can see that our user is **pi**, the default user for a Raspberry Pi. Next we can see **raspberrypi**, which is the name of the computer, often called the hostname. Next we see a



The **apt-get update** command generates a lot of text – here we can see the command contacting all of the servers that contain the Raspbian software. This will update the list of installable software.

tilde character "**~**", this is shorthand for that user's home directory, and it tells us where we are (by default we're located in our home directory). Finally we see a Dollar sign, which tells us that our user has no super powers.

Moving around

Lets try out a command that will tell us where we are. It's called Print Working Directory (**pwd**), and to use it we type.

\$ pwd

You will now see the absolute file path to your home directory – think of it as the full address for where you are.

To move around the system we need to know what is around us, and to do this we need to list the contents of a directory. In a terminal type

\$ ls

You'll see lots of text appear on the screen, but what is important right now is that we can identify files from directories. You can see **Desktop** in blue; this is a directory, so any text in that same colour also denotes directories. Let's go into the **Desktop** directory in a terminal type.

\$ cd Desktop

After pressing Enter we will move into the **Desktop** directory; to verify this, type in **pwd** and you should see **/home/pi/Desktop**. So we're now in the correct directory, take a look around using **ls**. To go up a level, by which we mean to navigate closer to the root of the file system, type

\$ cd ..

Now type in **pwd** –you should be back in your home directory. Let's try and move into a directory not inside our home. In a terminal type

\$ cd /dev

Now type in **pwd** and you will see that we are no longer inside our home, we are in the **/dev** directory. Take a look around using **ls** and you will see lots of files that represent devices, physical and virtual, that are attached to our Pi. This directory is full of stuff and right now we don't need to know too much about it so let's return to the safety of our home by typing:

\$ cd

Now that we're back in our home directory, let's create a simple text file in the terminal type

Sneaky sysadmin skills

Repeat the last command
You can press the Up arrow to reverse search through your commands (pressing Down goes forwards). Once you find the command, press Enter to action it.

See all of the commands that you have used
The **history** command records the commands entered in the terminal. You can type **history** into the terminal but it will whiz along really quickly. So, using a pipe (a tool to redirect the output of one command to become the input of another), we can pause the output and press the arrow keys to go through the list.

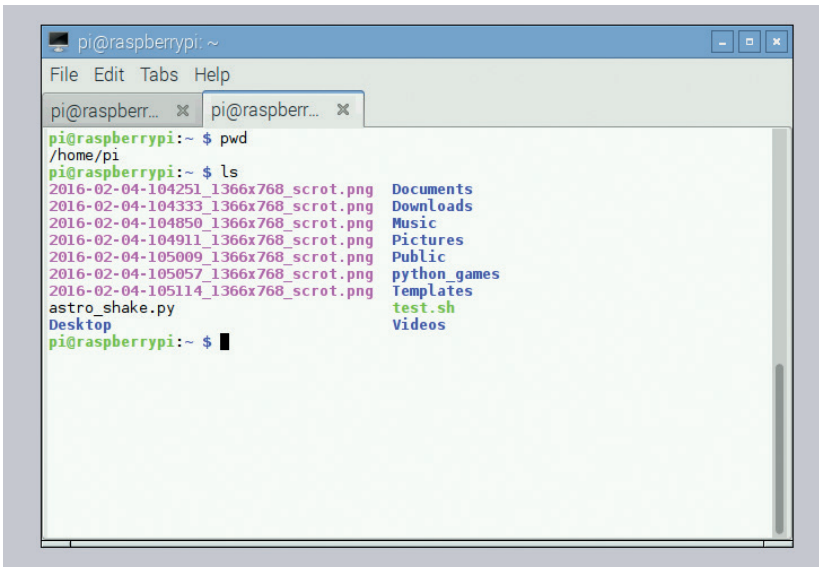
\$ history | less

Use a command from history
When we used the **history** command you may have noticed that the entries were numbered. These numbers can be used to run a command again, without typing it. All we need to do in a terminal is type an exclamation mark followed by the number of the command, for example.

\$!22

I forgot to run that command as sudo
We've all been there, we type in that command to change some aspect of the system, such as adding a new user to the system, but we forgot to add **sudo** to the start of the command. We pressed Enter and nothing happened. Fear not! To repeat the last command but with super powers all you need to do is type

sudo !!



We can use the **pwd** and **ls** commands to learn where we are in the file system and what is around us. These are basic building blocks to our Linux skills.

\$ nano hello.txt

You will now see a very basic text editor called *Nano*. We can use *Nano* to create and edit text files in any programming language, and we can edit system files on our Raspberry Pi. For this example we shall write a simple

Hello World!

So now let's save our work... but where is the save button? At the bottom of the screen you can see six columns of text, these comprise our menu. First we need to "WriteOut" our work, so press **Ctrl+O** to open the save dialog. You will be asked what filename you want to save the file as; we specified **hello.txt**. With our work saved we now need to exit *Nano*, so press **Ctrl+X** to exit and return to the terminal. If you type **ls** now you'll see the file that we have just created. Would you like to take a sneak peek inside the file without using *Nano*? In the terminal type

\$ less hello.txt

You will now see the contents of the file; to exit, press **Q**.

We've created a file, now let's create a directory called **stuff**. In the terminal type.

\$ mkdir stuff

Type in **ls** to see the new directory, now try to change the directory to the new **stuff** directory using **cd**. Once you've finished return to the home directory.

Copy and move files

Now we have a directory called **stuff** let's copy our **hello.txt** file into this new directory. In the terminal type

\$ cp hello.txt stuff/

The **copy** command is shortened to **cp** and creates an

identical copy of a file in the target directory, which in this case was **stuff/**. Take a look inside the **stuff** directory to verify that the copy has worked. Let's return to our home directory by typing **cd**. We shall create another file but this one shall be blank. To create a blank file we use the command **touch** followed by the name of the file, so in the terminal type.

\$ touch blank.txt

So now we have a file called **blank.txt** in our home directory, but we don't want it there, so let's move it into our **stuff** directory. Just like the copy command, the move command is shortened, this time to **mv**. In the terminal, type the following.

\$ mv blank.txt stuff/

Now change directory to **stuff** using **cd**, and then look inside using **ls**. Can you see the file? We can also use **mv** to rename a file and it works like this. In a terminal type:

\$ mv blank.txt renamed.txt

From the output of the **ls** command you should see that there are two files, one of which is called **renamed.txt**.

Delete files and directories

So now we have a directory called **stuff** which we are currently in, and it has two files inside of it. Let's do a little housekeeping and tidy up the directory. The command to delete files is called **rm**, and it should be used with great care. There is no easy undelete or recycle bin – once it is deleted, consider it gone forever. We'll remove one of the files, **renamed.txt**, like so:

\$ rm renamed.txt

Now use **ls** to look in the directory. You should see only one file now.

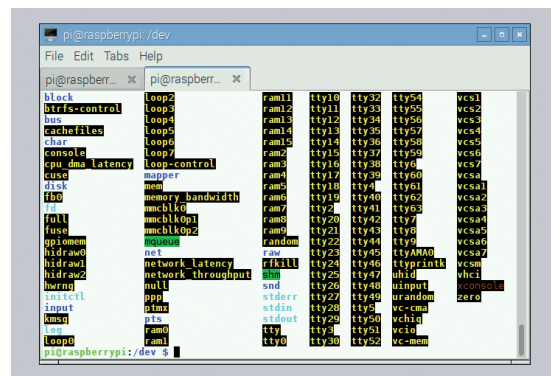
Let's delete an entire directory. Return to your home directory and type the following into the terminal.

\$ rmdir stuff

What happened? Did you get an error?

rmdir: failed to remove 'stuff/': Directory not empty

So what does this mean? Well we can only use the **rmdir** command on empty directories, so how can we delete directories and their contents? Well, the **rm**



The **/dev** directory is full of devices (virtual and physical) that we can use. Only skilled users with super powers should really do any work in here.

Controlling your Raspberry Pi over a

Sometimes we cannot be sitting in front of our Raspberry Pi, either we are working away or we need to remote control our Pi. Linux comes with SSH, a secure shell, that enables remote control in a secure manner. The Raspberry Pi comes with an SSH server ready to go – all we need to do is connect our Pi to the network and power it on. From our PC we can then use an SSH client to log in. For Linux users we have this built in to the terminal, as do Mac users. But for Windows you will need to download an application called *PuTTY*: http://portableapps.com/apps/internet/putty_portable.

For each operating system we need to know the username (normally `pi`) and our password (the default is "raspberrypi").

For Linux and Mac users, open a terminal and type

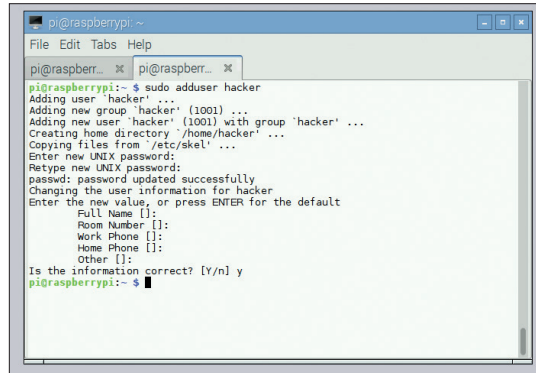
```
ssh pi@raspberrypi.local
```

For Windows users, install and open *PuTTY*. In the new Window you will see "Host Name". Type in:

```
raspberrypi.local
```

Windows users should click on Open, and you'll be asked for your username, which is "pi".

For all operating systems you will be asked for your password. Press Enter and you are now in control of your Raspberry Pi, so any command issued in the terminal will be processed by the Pi and the output sent to your screen.



provide the details of that user, name, phone number etc. This is not mandatory, so feel free to press Enter to bypass. The last question asks if the information is correct, if so press Y then Enter.

Our new user, **hacker**, needs to have super powers, so without any radioactive spillages or elaborate backstory we shall give them the power they need. It's rather easy to add a user to a group; we'll use the same command as before but add the group name at the end of the command. In a terminal type the following.

```
$ sudo adduser hacker sudo
```

So now we can test this user by switching our user to them. In a terminal type

```
$ su hacker
```

You will need to use the password that we created earlier. Once this is entered you will see the terminal change to reflect that you are now logged in as the user **hacker**. So can **hacker** use **sudo**? Let's try by repeating the **ifconfig** command that we used earlier. In a terminal type:

```
$ sudo ifconfig
```

You should be able to see the network connections on your Pi plus lots of other text, if you see a rather stern note that you have been reported to root for trying to use **sudo**, then repeat the process to add **hacker** to the **sudo** group.

What's my IP address?

Sometimes we need to quickly know the IP address of a computer without faffing around with a GUI. We can use two commands: if you just want the IP address, type the following into a terminal.

```
$ hostname -I
```

But if you really need to know everything about a connection, for example the IP address for your Wi-Fi dongle, Ethernet connection and their MAC addresses then you can use.

```
$ ifconfig
```

Congratulations: you have taken your first and rather important steps with the Linux command line. Keep these skills sharp and they will help your future hacks.

command can be used with directories, so to delete **stuff** type the following.

```
$ rm -rf stuff/
```

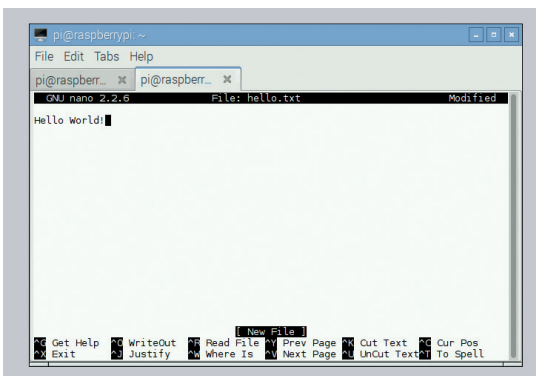
The directory **stuff** and its contents will now disappear. You can use **ls** to look for it but you will never find it.

Create users

On a typical Raspberry Pi we have one user, **pi**, but what if we shared our Raspberry Pi with another user? Do we want them to access our work? Well the answer is clearly no, so let's create another user using the command **adduser**. In a terminal type

```
$ sudo adduser hacker
```

Now we used **sudo**, as creating a new user is a system-wide activity. This command creates a new user called **hacker** and then proceeds to ask you to create a password for this user. It next asks you to



Nano looks really basic but it hides a really powerful interface for working with text and files in the terminal. There are other editors such as *Vi* and *Emacs*.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

The **adduser** command is rather helpful and asks us lots of questions in a steady manner.

GET STARTED WITH OPENSTACK AND JUJU

You don't need a Google-sized data centre to play with the revolution in the cloud.

GRAHAM MORRISON

WHY DO THIS?

- Create your own cloud
- Install Mediawiki (an more) with Canonical's JuJu

TOOLS REQUIRED

- Any PC from the last 5 years
- More than 12GB RAM
- 2–3 hours for the installer

Despite the continued hype, 'cloud computing' isn't necessarily all that new. It's very similar at a hardware level to what has always been found in data centres: a collection of storage, networking and computing resources. The difference is in the way those resources are managed and used.

Cloud infrastructure is very elastic – you don't need to hog a single processor, Ethernet connection or hard drive. Instead, applications in a cloud can typically scale from tiny to huge, and they can scale dynamically according to demand, whether that's more storage, RAM or computing power. The

applications themselves don't exist on a single machine either, but rather on an abstraction of those resources, effectively an OS for the cloud.

OpenStack is perhaps the largest of these 'operating systems'. It's a stack of open source software designed to control pools of resources, and it's being used everywhere, from PayPal and eBay, to CERN and NASA. The difficulty has always been finding a way to play and install OpenStack yourself, when you don't have gargantuan computing resources at your disposal. Fortunately, if you've got a powerful enough machine and Ubuntu 14.04, you now can.

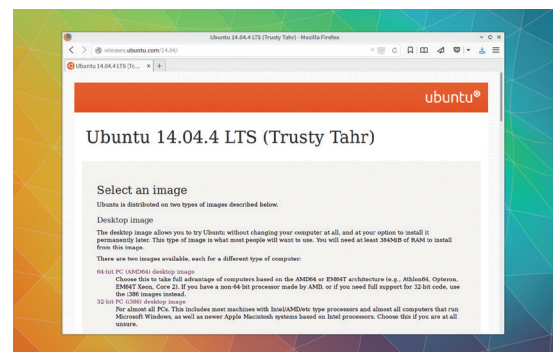
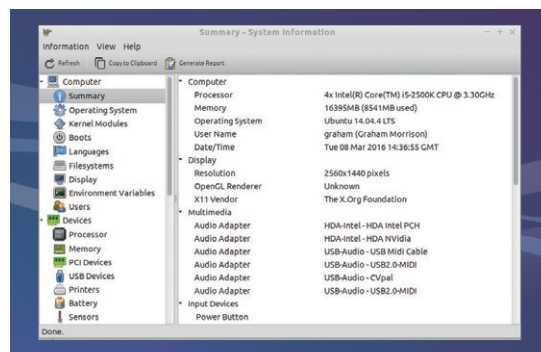
STEP BY STEP: INSTALLING AND RUNNING OPENSTACK

1 Find the hardware

The biggest problem when it comes to playing with OpenStack is having a stack of computers handy. Fortunately, to test out the technology, all you need is a reasonably powerful single machine that's capable of installing Ubuntu. You'll also need a multi-core CPU and lots of RAM. For reference, the machine we used has 16GB of RAM and an Intel core i5-2500K CPU from 2011. More RAM will help because we're effectively going to create an OpenStack installation running across more than 10 Linux Containers (virtual machines) on a single PC. Each one of those containers will be asking for several GB of RAM, most of which they won't need but your system will still grind to a halt while OpenStack works out the RAM allocation. An SSD could help here, but our system eventually made it with ordinary spinning storage and several cups of tea.

2 Install Ubuntu

The method we're going to use to install OpenStack leans heavily on the work of Ubuntu and Canonical. We're going to use their pre-built packages and installers, as well as Canonical service orchestration tool, JuJu. JuJu can be thought of as being a little like a package manager for cloud applications, enabling you to install (deploy, in cloud terminology) and tie together services with just a few commands, very much like **apt**. Of course, there's a lot more going on in the background, but the end result is that JuJu enables you to easily add services to your own cloud after you get it running. JuJu is even used by the installer to create the OpenStack configuration. All of which is a long way of saying you'll need to install Ubuntu, and it works best if you can give it a fresh new installation. We had best compatibility results with 14.04 LTS, although 16.04 beta worked too.



3 Add the packages

With Ubuntu installed, we're going to use the 'Ubuntu Openstack Installer' to grab all the latest packages and dependencies. The current focus for these packages is Ubuntu 14.04, because of its long term support status. But as 16.04 is also a long-term release, and imminent at the time of writing, it may be a better option. We did try the packages running on a beta of 16.04 and they worked, but we'd still safely recommend 14.04 unless you've got a hardware requirement only satisfied by a newer release. Open a terminal and type the following to add the **cloud-installer** package repository and install the dependencies we found we needed:

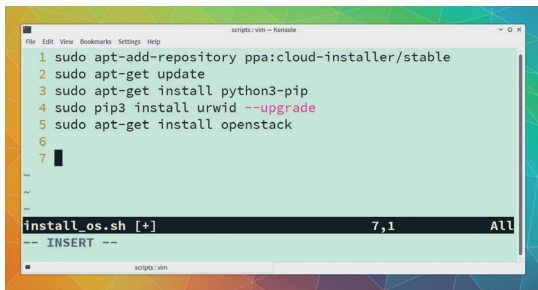
```
sudo apt-add-repository ppa:cloud-installer/stable
```

```
sudo apt-get update
```

```
sudo apt-get install python3-pip
```

```
sudo pip3 install urwid --upgrade
```

```
sudo apt-get install openstack
```



4 Start the installation

Installing the dependencies is only the first step. The time-consuming part comes from running the installer and waiting for it to build a workable environment on your system. To begin the process, type the following:

```
sudo openstack-install
```

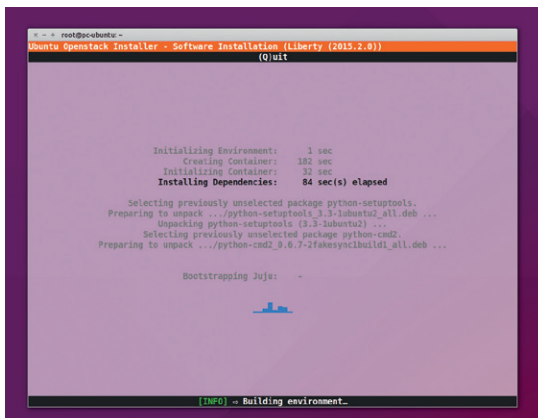
The interface to this installer runs via *curses* on the command line. We're running version 0.99.27 of the installer (March, 2016), so there's a small chance that something may look different in a more recent version, but the process should be identical. The first question you'll be asked is what type of installation you'd like to perform, and you need to choose the 'Single' option at the bottom. As the text describes, this is the option for installation on a single machine.



5 Put the kettle on

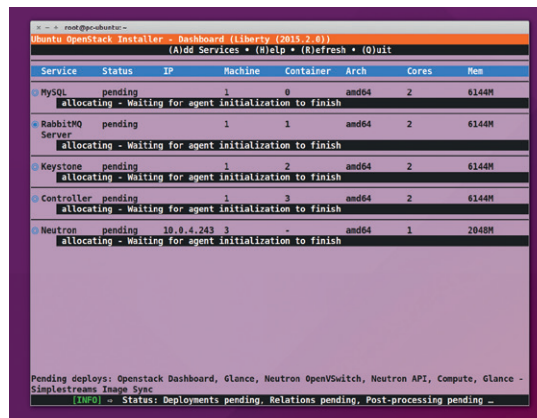
After being asked for the installation type, you'll next be asked for a username and password. This username and password will be used to access the account and the dashboard, which is the OpenStack module used to provide a web management interface. The first stage of the installation will appear as in the screenshot, as the installer will download and pack all the various tools required.

OpenStack is built with Python (which is why we made sure **pip** was installed earlier), and a lot of the requirements will be for Python packages and the creation of the first container. After that, you should see the 'Bootstrapping Juju' message, which should kick off the second stage of the installation.



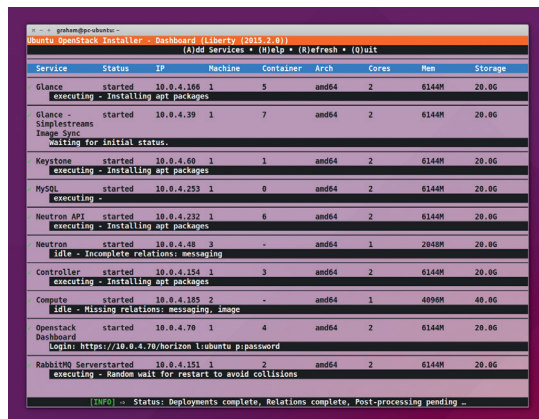
6 Make a cup of tea

In total, we waited over an hour for this step to complete, although subsequent installations were a lot quicker. This is the part where new virtual machines (containers) are spawned and slotted into the OpenStack ecosystem. Each container is needed to perform a different task or run a different OpenStack module, and they'll all end up talking to one another to create the final configuration. Each will appear gradually on the terminal and go through a series of stages after starting off with 'Pending,' and it's this step that will test your machine's performance to the maximum. We'd recommend spending the time with a nice cup of tea and reading a little about what each module does in OpenStack.



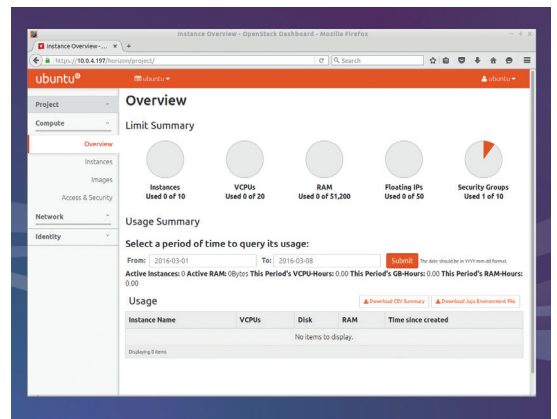
7 Nearing completion

From top to bottom, the modules that OpenStack instantiates are Glance, Keystone, MySQL, Neutron/API, Controller, Compute, Dashboard and RabbitMQ. Briefly, Glance is used to discover and register disk and server images. Keystone is the identity and authentication service. Neutron handles the virtual networking. The 'controller' orchestrates everything, the 'Compute' module actually runs the instances on the hardware and the RabbitMQ server handles the messaging. There's also the dashboard, which provides a web administration interface to the cloud. When the install has finished, the [INFO] text at the bottom will display 'Status: Deployments complete, Relations complete, Post-processing complete.'



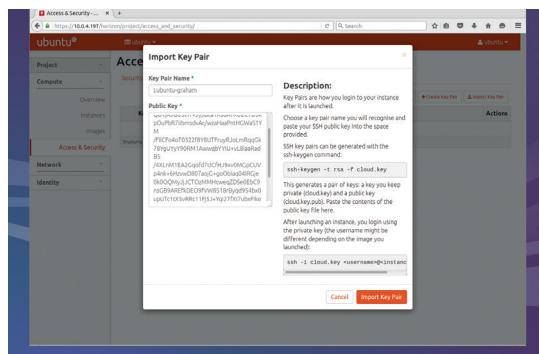
8 Horizon/Dashboard

On the same machine, enter the login URL. This link will be an HTTPS connection, and you'll need to ask your browser to make an exception for the self-signed SSL certificate before you can continue. You'll then be asked for the username and password, as shown in the installer terminal, which is now simply showing the output from the **openstack-status** command. The dashboard defaults to an overview of the instances, virtual CPUs, RAM units, IP addresses and security groups being used by your cloud. There's nothing useful to see because we haven't added any of our own instances, but you can do this by switching to the 'Images' page of the Compute module, which should list a single image name. First, we need some security.



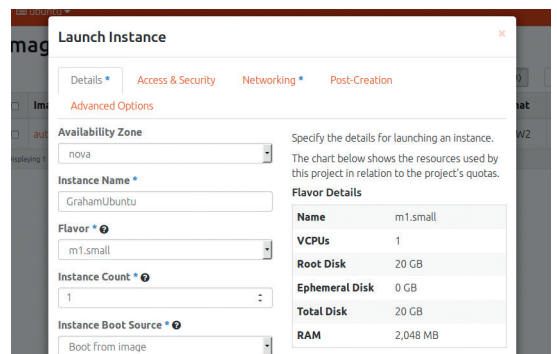
9 Set up security

Security is paramount when it comes to the cloud, and before you'll be able to connect to any virtual machines you'll need to create an SSH keypair. This can be done from the command line by typing **ssh-keygen -P "" -b 2048**, but don't let it overwrite your current files. Put the new files somewhere safe. In the dashboard, go to the Compute > Access & Security page, click on Key Pairs and use the Import Key Pair button to open a panel where you can paste the contents of the **id_rsa.pub** file you just created. Give the key a name and save it. You now need to enable SSH access through OpenStack's firewall. Click on the Security Groups tab followed by Manage Rules. Click on Add Rule and select SSH, making sure Remote is set to CIDR. This will allow external SSH connects to find a route to your new instance.



10 Spinning up your own instance

By default, you should see an image for the Ubuntu version you're running on the Compute > Images page. New images can be added using the Create Image button, but for this example we're going to spin up the image already listed. This is as simple as clicking on the Launch Instance button. You need to give your instance a name, select a Flavour (start small) and choose your fresh security key from the Access & Security tab. When you're happy with the options, click on Launch. The instances list will appear with a 'Spawning' status in the task field. Your new virtual machine is now being created and within a few moments – not an hour this time – 'Spawning' will change to 'Running' and your cloud will now be hosting your very own instance, just waiting for you to connect to it and launch a startup business.

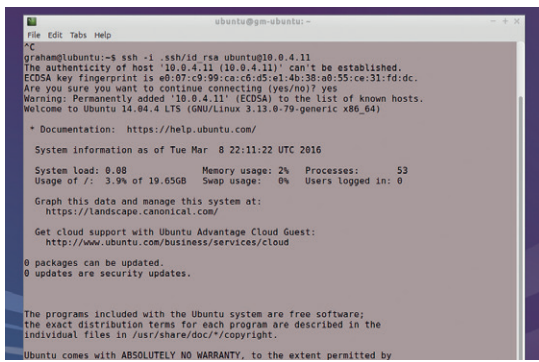


11 Connecting to your new instance

Before we can connect, we need to attach the instance to an IP address accessible outside the cloud. You can do this by selecting 'Associate Floating IP' from the drop-down Action menu. After a simple panel, the floating IP address will appear in the IP address field, and you should be able to SSH into your new machine by typing the following, replacing **id_rsa** with the path to the private key we generated earlier:

```
ssh -i id_rsa ubuntu@floating_ip
```

After accepting the SSH fingerprint, SSH should automatically connect you to your fully fledged instance running on your very own cloud, and you can install packages and do whatever you need just as you would a real machine on a real server. Try installing *Apache 2*, for example (and don't forget to allow HTTP connections).



12 Connecting to JuJu

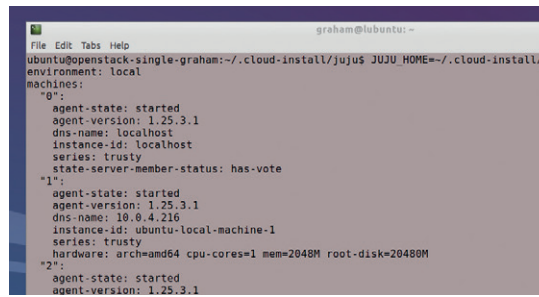
Building your own applications running on top of your own containers is obviously a very powerful step, but there are other options that cut out manual labour. Canonical's JuJu is one of these, and it's already being used by your system to deploy the various OpenStack modules on your machine. You can see JuJu running by connecting to the container that's currently orchestrating OpenStack by typing the following:

```
sudo lxc-attach -n openstack-single-$USER
```

```
su ubuntu
```

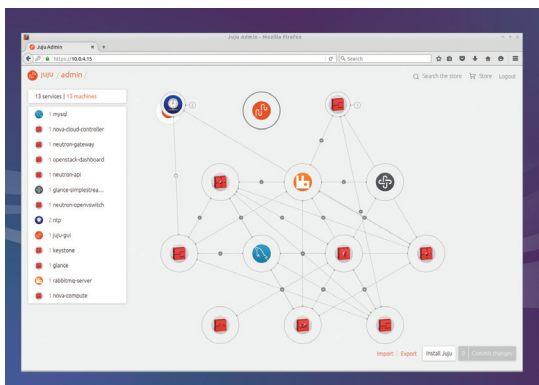
```
% JUJU_HOME=~/.cloud-install/juju juju status
```

The final command, **juju status** will produce output very similar to the **openstack-status** view we used during installation, only this time JuJu is the tool doing all the hard work. Working with JuJu on the command line is straightforward, but there's also a wonderful web interface that can be installed by typing **juju deploy juju-gui** followed by **juju expose juju-gui**.



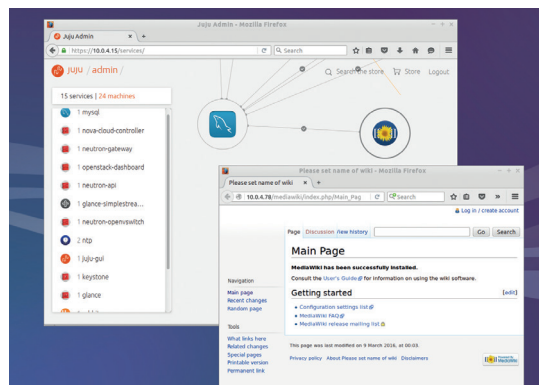
13 JuJu GUI

You should be able to see from the installation status terminal that JuJu GUI has been added as a new service, usually at the bottom of the list where it will include a login URL. Enter this into a local browser with the usual HTTPS caveat and use **admin** and your password to enter the sandbox. The site that opens is a really impressive graphical overview of your current cloud, as orchestrated by JuJu. You can see all the installed services and how they're connected. You can even drag them around and their elastic connections will follow. You can also use the list of services on the left to isolate specific modules and see how they're interconnected. It's a great way of learning how OpenStack is configured, but the real magic comes from clicking on the 'Store' button in the top-right.



14 Installing Mediawiki

The JuJu store is like a package repository for cloud applications. Some may need connections to be made between resources after installation, which you can do in the GUI, but you can get around this by installing an all-in-one bundle. To try the installation out, search for **mediawiki**, select it and click on 'Add To Canvas'. You'll see it appear in the main view (canvas). It needs to be connected to MySQL by dragging from the MySQL icon to Mediawiki and selecting **mysql:db mediawiki:db** as the relation. Click 'Commit Changes' followed by 'Deploy'. Finally, select Mediawiki and click 'Expose' followed by 'Commit' to make the server live. Wait for the initialisation to finish and 'juju status mediawiki' to check its status and get the IP address after initialisation.



ILLUMINATE YOUR LIFE WITH LINUX PART 3

Take your next steps in home automation by using Linux to control your lights.

MARK CRUTCH

WHY DO THIS?

- Control the colour of your lights
- Learn about data-* attributes
- Mix HTML and SVG

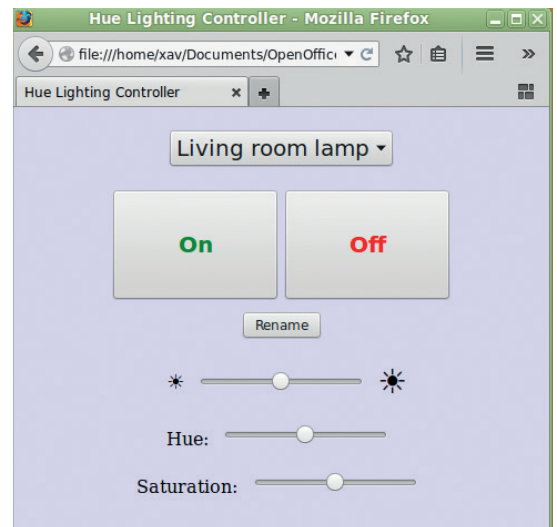
Over the past couple of issues we've been exploring ways to control Philips Hue and Lux "smart" bulbs using a Linux box. Last time we left you with a simple web page that would let you turn a light on and off, change its name and adjust its brightness. This time we're going to add some colour controls for Hue bulbs.

You could base your new code on the brightness slider we used last time, but it's not a simple copy-and-paste job. The API requires the hue value to be specified in a range from 0 to 65535. That's easy enough: just change the min and max attributes on the new slider. The saturation value runs from 0 (white) to 254 (100% colour), but we prefer to have our slider running in the opposite direction, so that white is at the right-hand end of the scale. There are a few ways that we could flip our scale to accommodate this requirement, but we chose to just change the min and max to run from -254 to 0, then negate the value as it's passed to our new `set_saturation()` function. This is the code we added to our HTML:

```
<div id="div-hue">
  <label id="label-hue" for="hue">Hue:</label>
  <input id="hue" type="range" min="0" max="65535"
    onchange="set_hue(this.value);" />
</div>
<div id="div-saturation">
  <label id="label-saturation"
for="saturation">Saturation:</label>
  <input id="saturation" type="range" min="-254" max="0"
    onchange="set_saturation(-this.value);" />
</div>
```

And here's the extra pair of JavaScript functions we call when these sliders are moved:

```
function set_hue(nValue)
{
  var sURL = `${sBaseURL}/lights/${$("lights").value}/state`;
  var sPayload = `{"hue":${nValue}}`;
  fetch(sURL, {method: "PUT", body: sPayload});
}
function set_saturation(nValue)
{
  var sURL = `${sBaseURL}/lights/${$("lights").value}/state`;
  var sPayload = `{"sat":${nValue}}`;
```



Our first pass at adding colour controls works, but looks a little plain.

```
fetch(sURL, {method: "PUT", body: sPayload});
}
```

If you've got a mixture of Hue and Lux bulbs it would be nice to hide the additional UI for those bulbs that only support brightness changes. There are a few steps needed to achieve this: we need a way to show and hide the relevant parts of the interface; we have to store some extra information to indicate which bulbs can change colour; and we need a function that fires whenever the currently selected bulb is changed.

In the HTML page we decided to wrap our new controls in another `<div>`. We've given ours an ID of `div-colour`, and enclosed the `div-hue` and `div-saturation` sections inside it. This gives us a single target to show and hide in order to suppress both controls at once. The `<select>` element also gets a new `onchange` attribute with a value of `check_lamp_type()`; – this will be the name we use for our new function. To hide the new `<div>` we'll set or remove a "hidden" class, and create a line in our CSS file that tells the browser what to do when the class is present:

```
.hidden { display: none; }
```

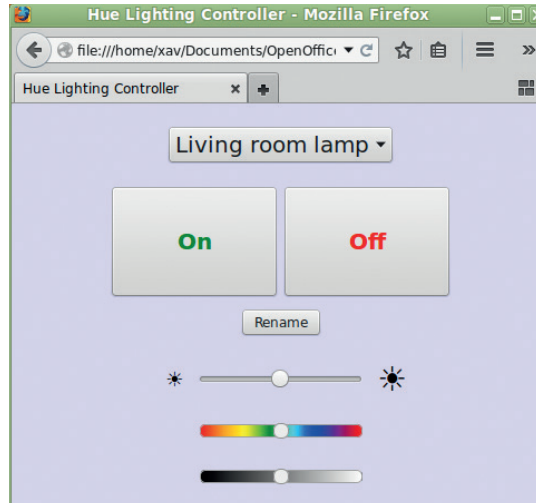
We now have the capability to hide the controls, but we still don't have any information about which lights need them shown and which ones don't. It's time to extract a little more information from our initial query

to the Hue API in order to set a flag for those lights that support colour. We'll determine this by testing for the presence of a **hue** property on the **state** object in the JSON. Modify the **for** loop in your **main()** function to look like this:

```
for (var sLightID of Object.keys(oJSON)) {
    var oOption = document.createElement("option");
    oOption.setAttribute("value", sLightID);
    if (oJSON[sLightID].state.hue !== undefined) {
        oOption.setAttribute("data-lamp_type", "colour");
    }
    oOption.innerHTML = oJSON[sLightID].name;
    oSelect.appendChild(oOption);
}
```

The only addition here is that, when we create the **<option>** elements inside our popup menu, we're also adding a **data-lamp_type** attribute to any bulbs that have a **hue** property in the JSON. As we're only dealing with two possibilities here – colour or not colour – we're just using the presence of the attribute to distinguish between them. Nevertheless, it makes sense to give the attribute a sensible value to make the code more readable.

We could simply have called the new attribute **lamp_type** and browsers would have been perfectly happy. But it's possible (though unlikely) that some future revision of HTML could introduce a **lamp_type** attribute to the standard that conflicts with our own version. To avoid problems like this, HTML 5 has a rule whereby you're safe to use your own attribute names provided they're prefixed with **data-**. This becomes a kind of semi-private namespace that the HTML working group promises never to pollute with their own additions. These **data-*** attributes can be read and written using the normal **getAttribute()** and **setAttribute()** methods, as we've used in the **main()** code, but they're also accessible via a "**dataset**" property on your element, in which case the **data-** prefix is dropped in the JavaScript, making things a little more readable. You can see an example of this in the new **check_lamp_type()** function that will tie all



By adding some CSS, our colour controls actually have some colour!

these new additions together:

```
function check_lamp_type() {
    var oOption = $("lights").selectedOptions[0];
    if (oOption.dataset.lamp_type === "colour") {
        $("div-colour").classList.remove("hidden");
    } else {
        $("div-colour").classList.add("hidden");
    }
}
```

First we get the currently selected option, then the next line checks the state of our new attribute, via the **dataset** property, and adds or removes the class as necessary. Reload your page, and you should now find that the Hue and Saturation controls are shown and hidden based on the type of bulb. This approach of pulling data from the JSON and storing it in a **data-*** attribute is one method that could be used to keep the controls in sync with the current state of the selected bulb, if you want to create a more sophisticated interface.

The colour controls are visible unless we explicitly hide them, but that only occurs when a new bulb is chosen from the popup menu. What happens if our first bulb happens to be a Lux bulb? We would

PRO TIP

Download the example code from www.peppertop.com/hue.zip

Working with RGB

One omission in the Hue API is the ability to set a colour using RGB values. This requires the developer to do some additional work to convert an RGB value to hue, saturation and brightness, though the mapping isn't a simple one. We prefer to keep the brightness control separate from the hue and saturation, so here is a cut-down function for just returning those components for a Hue bulb, when given red, green and blue values in the range 0–255:

```
function rgb_to_hs(r, g, b) {
    var h, s, c;
    r = r/255, g = g/255, b = b/255;
    var minRGB = Math.min(r,g,b);
    var maxRGB = Math.max(r,g,b);
    c = maxRGB - minRGB;
    if (c === 0) return [0, 0];
```

```
    if (maxRGB === r) {
        h = ((g-b)/c) * 60;
    } else if (maxRGB === g) {
        h = ((b-r)/c + 2) * 60;
    } else {
        h = ((r-g)/c + 4) * 60;
    }
    h = (h < 0) ? h + 360 : h;
    // h is 0-360 degrees; map it to 0-65535 for Hue API
    h = Math.floor(h * 183);
    s = Math.floor((maxRGB - minRGB)/maxRGB * 254);
    return [h, s];
}
```

The HTML 5 colour picker widget (**<input type="color" />**), returns its value as a hexadecimal number in the form **#rrggbb**. To use this with our

conversion function requires us to split the string into its hexadecimal components, then convert those to decimal. Here's a function that you can hook into the picker's **onchange** handler that does the hard work for you using a regular expression and a few **parseInt()** calls:

```
function colour_from_picker(sHexRGB)
{
    var aValues = sHexRGB.match(/^#(..)(..)(..)$/);
    var r = parseInt(aValues[1], 16);
    var g = parseInt(aValues[2], 16);
    var b = parseInt(aValues[3], 16);
    var [h, s] = rgb_to_hs(r, g, b);
    set_hue(h);
    set_saturation(s);
}
```

be showing the colour UI, even though it's not appropriate. The fix is quite simple: we simply need to call our new function once we've finished initialising the UI. Just add a line to call `check_lamp_type()` after the `for` loop in the `main()` function, before the closing brackets of the second `then()` method.

Improving the UI

One issue with our user interface at the moment is that there's no indication as to what colour any particular value for the hue slider will produce. We know it's red at either end, green a third of the way along, and blue at two thirds, but it would be better to have a graphical representation of that. It is possible to style HTML sliders, but the syntax is a little different for each browser; however, we only want to add some colour to the "track" of the slider, and only for *Firefox* and *Chrome/Opera*, since our JavaScript is too cutting-edge for other browsers anyway. This results in a relatively small amount of code to add to the CSS:

```
#label-hue { display: none; }
#hue::-moz-range-track {
  background: linear-gradient(90deg,
    red,orange,yellow,green,cyan,blue,purple,red);
  border: 1px solid darkgrey;
  height: 10px;
  border-radius: 5px;
}
#hue::-webkit-slider-runnable-track {
  -webkit-appearance: none;
  background: linear-gradient(90deg,
    red,orange,yellow,green,cyan,blue,purple,red);
  border: 1px solid darkgrey;
  height: 22px;
  border-radius: 5px;
}
```

We also added the same code a second time, but modified for the saturation control. Just replace every instance of "hue" with "saturation", and instead of a linear-gradient running through the colours of the

rainbow, it just goes from black to white. With both sets of additions in place, our UI is starting to look almost professional:

HTML range inputs are one-dimensional widgets. They enable you to change a single value by moving a slider along a fixed track. Another option, when you have two related parameters,

is a two-dimensional input widget that lets you click anywhere within a plane in order to select a pair of coordinates.

There are several different ways to approach this, each with their own pros and cons, but we've chosen to use SVG to draw the UI for our hue-saturation selector. This is an XML-based language for drawing vector graphics, and has been a W3C standard for a long time. Its take-up has been very slow, largely due to lack of support in Internet Explorer, but the last few years have seen a surge in its popularity, resulting not only in IE gaining support, but also in SVG becoming

a first-class citizen in HTML5. This means that you can throw a block of SVG code into any HTML5 web page without having to concern yourself with the complexities of combining two XML languages in a single document. Add the following SVG to your HTML file, just before the closing tag of the "div-colour" section (so that the new code is inside the <div>, just after the hue and saturation sections):

```
<svg id="svg" width="100%" height="100%"
  viewBox="0 0 65535 254"
  preserveAspectRatio="none">
<defs>
  <linearGradient id="gradient-hue">
    <stop stop-color="red" offset="0%" />
    <stop stop-color="orange" offset="14%" />
    <stop stop-color="yellow" offset="28%" />
    <stop stop-color="green" offset="42%" />
    <stop stop-color="cyan" offset="56%" />
    <stop stop-color="blue" offset="71%" />
    <stop stop-color="purple" offset="85%" />
    <stop stop-color="red" offset="100%" />
  </linearGradient>
  <linearGradient id="gradient-saturation"
    x1="0" y1="0" x2="0" y2="1">
    <stop stop-color="white" stop-opacity="1"
      offset="0%" />
    <stop stop-color="white" stop-opacity="0"
      offset="100%" />
  </linearGradient>
</defs>
<rect x="0" y="0" width="100%" height="100%"
  fill="url(#gradient-hue)" />
<rect x="0" y="0" width="100%" height="100%"
  fill="url(#gradient-saturation)"
  onclick="svg_colour_selected(evt);" />
</svg>
```

That's a lot of code, but what it does is actually quite simple: it draws two rectangles, one on top of the other, each filled with a different linear gradient in order to provide a sweep of colours in one direction and saturation in the other. Let's look at it bit-by-bit to get a better understanding of what's happening.

We start with our `<svg>` container. This tells the browser that we're introducing a block of SVG code into the page, and that it should take up 100% of the available width and height in its parent container (the `div-colour` block). That lets us adjust the size of the parent in our CSS file, knowing that the SVG will automatically resize to match. Within a block of SVG it's possible to introduce a different coordinate system to that used by the main page, using the `viewBox` attribute. This takes the x and y coordinates of the top-left and bottom-right corners of the new coordinate system, allowing us to specify that, within the SVG world, our coordinates run from 0 to 65535 in the x-direction and 0 to 254 in the y-direction, regardless of the "real-world" dimensions of the web page. The `preserveAspectRatio` attribute ensures that our rectangles will fill the available space, even though it means ignoring their native aspect ratios.

PRO TIP

If you can ssh into your home network, use the `-R` option to forward port 80 from your web server to a port on the local machine, allowing you to control your lights from anywhere in the world.

Inside the SVG we have a `<defs>` section to hold definitions for things we'll need later in the code. There are a pair of linear gradients, each containing `<stop>` elements that define the colour stops. Unfortunately SVG doesn't automatically space the stops for you, so you have to specify the offset for each one. The `x1`, `y1`, `x2` and `y2` values on the saturation gradient ensure that it runs from top to bottom, rather than the default of left to right. Notice also that the saturation gradient runs from white to white, but with the opacity running from 1 to 0.

Finally we get to the rectangles we're drawing. They're both positioned at the top-left corner of our coordinate space, and extend to 100% of it in either direction. Each is filled with a gradient from the `defs` section by a reference to its ID. The "painters model" that SVG uses means that later objects are drawn on top of earlier ones, so our second `<rect>` is placed on top of the first. Because the saturation gradient is partially transparent, the hue gradient will show through from the rectangle below. As it's on top, the second rectangle is also the right place to put our click handler – the first one never receives any clicks itself.

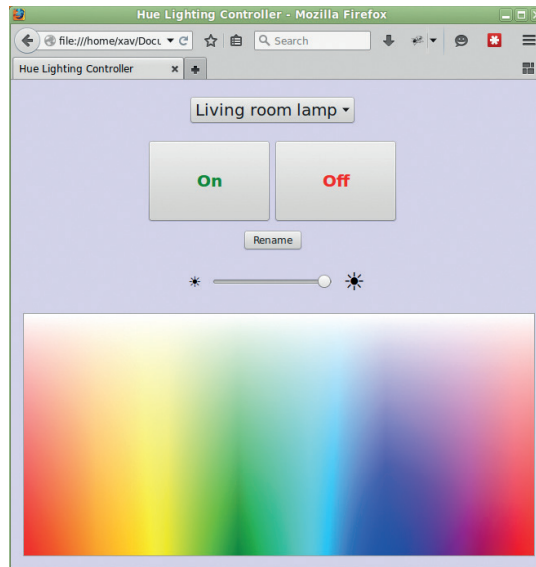
If you reload the page at this point you'll be disappointed to see the new selector only appears as a thin strip across the screen. This is because we haven't fixed the height of the parent `<div>`, so the SVG collapses down to the minimal amount of vertical space it can. Let's hide the old sliders and give the new control some breathing room with this addition to the CSS file:

```
#div-hue { display: none; }
#div-saturation { display: none; }
html { height: 95%; }
body { height: 100%; }
#svg { border: 1px solid darkgrey; }
#div-colour {
  height: 50%;
  margin: 0;
}
```

By setting the `<html>` to 95% we allow the page to fill most of the window, but not enough to cause scroll bars to appear. The `<body>` is set to use all that space, allowing plenty of room for our `div-colour` to fill. We've set that to a percentage height, allowing it to grow and shrink as the window size changes.

The last step is to add some JavaScript to handle clicks within the new control. The `onclick` handler on the second rectangle passes the click event to `svg_colour_selected()` so let's add that, and another helper function, to the JS file:


```
function svg_colour_selected(e)
{
  var oSVG = e.target.nearestViewportElement;
  var [h,s] = convert_coordinates(oSVG, e.clientX, e.clientY);
  set_hue(h);
  set_saturation(s);
}
function convert_coordinates(oSVG, x, y)
```



Throw in some SVG and things really start to look fancy.

```
{
  var oPoint = oSVG.createSVGPoint();
  oPoint.x = x;
  oPoint.y = y;
  var oMatrix = oSVG.getScreenCTM().inverse();
  oPoint = oPoint.matrixTransform(oMatrix);
  return [Math.floor(oPoint.x), Math.floor(oPoint.y)];
}
```

In an ideal world we would be able to pull our SVG coordinates straight out of the click event. The x-coordinate could be passed straight to `set_hue()` and the y-coordinate to `set_saturation()`. Unfortunately the browser only gives us the values transformed into the HTML coordinate system. The transformation takes place through a matrix multiplication, and it's possible to query the `<svg>` element for the specific coordinate transformation matrix that was used via the `getScreenCTM()` method. The returned matrix has an `inverse()` method that gives us a matrix for transforming values in the opposite direction. By creating an SVG point, setting its coordinates to the x and y values from the click event, then performing an inverse matrix multiplication, we can get the original SVG coordinates back once more.

With our SVG-powered colour selector working, that concludes our short foray into the world of smartbulbs. But we've only just scratched the surface. The web interface has plenty of scope for improvements or new features, and the Bash and Python examples from part one offer a wealth of possibilities for hooking the lights up to other data sources. As an unobtrusive notification system, your own house lights are hard to beat, so let us know on the forums if you come up with any particularly good ways to use these devices. 

PRO TIP

Another approach to laying out the web UI is to use the new CSS flexible box model, which makes it easy to create widgets that grow and shrink as the page size changes.

Mark Crutch now has his Hue lights flashing around a spinning office chair, to recreate his childhood dream of becoming Joe 90.

WRITE AN IMAGE CROPPING PROGRAM WITH GO

Learn the ropes of Google's excellent Go programming language.

AMIT SAHA

WHY DO THIS?

- Get to grips with one of the languages of the moment.

Go (or Golang) is attractive because of its C-like syntax, static typing (but with a feel of being dynamic typed) and its rich standard library. In this article, our aim is to introduce you to some of the features of Go. We do not assume that you know Go, but assume familiarity with another programming language. By the end of this article, you will have written a command line program that can crop images to your desired height and width. More importantly, you will have had a very short introduction to the language and how to use packages from the standard library as well as third-party packages. This will hopefully inspire you to go ahead and learn more about it. Let's get started!

Before we can write our first program, we will need to install the Go tools, which includes the Go compiler and other useful bits and bobs. On Fedora 23, we can install Go 1.5 using the distro's package manager:

```
$ sudo dnf -y install golang
```

On other Linux distros, please download the Linux binary and follow the instructions on the install page at <https://golang.org/doc/install>. Once the installation steps are completed, open your favourite terminal emulator and type **go version** and it should print a message similar to below:

```
$ go version
go version go1.5.2 linux/amd64
```

Functions in Golang are created using the **func** keyword, followed by the function name and a list of arguments

Now, we are ready to set up our workspace. Create a sub-directory called **golang** in your home directory (**/home/<user>**) and a sub-directory, **src**, inside it. All the source code for Go programs will live in the **src** sub-directory. This includes the programs we write as well as the third-party packages we will use (more on this later).

The directory tree for your workspace should look as follows:

```
golang/
|-- src
```

The go compiler and other tools expects the **GOPATH**

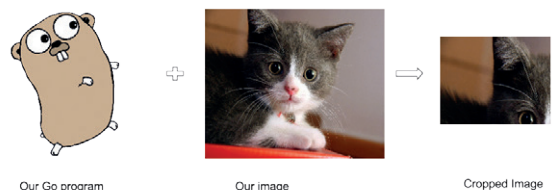


Illustration of the program we will write by the end of this article. (The Go gopher was designed by Renee French. (<http://reneefrench.blogspot.com>)).

environment variable to point to the workspace directory, so set the following to your **.bashrc** or the file relevant to your shell, so that it is always set when you start a new terminal session (Replace **<user>** with your username):

```
export GOPATH=/home/<user>/golang
```

Once you have set the above, start a new terminal session and type in

```
$ go env GOPATH
/home/<user>/golang
```

If you see your **GOPATH** printed correctly similar to above, we're all set to perform one final step before our first program.

Execute the command **mkdir -p github.com/linuxvoice/helloworld** while in the **src** sub-directory we created above. The directory tree should look like this:

```
-- src
  |-- github.com
    |-- linuxvoice
      -- helloworld
```

The reason we do this is that since all the Go code you write (or that of the third-party packages we use) will live in the same **src** sub-directory, it is important we have distinct sub-directories where they live. It is customary for the top-level sub-directory to be of the same name as your code hosting site (here, **github.com**), the next sub-directory to be the username (here **linuxvoice**), and finally the sub-directory for our code itself (**helloworld**).

Hello World

The program below will print the customary "Hello World" message when run. Type in the following program to a file **hello.go** and place it in the

helloworld sub-directory we created above:

```
// This is a simple program which prints Hello World
package main
import "fmt"
func main() {
    fmt.Println("Hello World!")
}
```

Your **src** directory should look as follows:

```
-- src
  -- github.com
  -- linuxvoice
  -- helloworld
  -- hello.go
```

Since Go is a compiled language, we first have to compile it. To do so, we will use the **build** sub-command of the **go** tool from the **src/github.com/linuxvoice/helloworld** directory:

```
$ go build hello.go
```

You should now see that a **hello** executable has been created, which we can execute as follows:

```
$/hello
```

```
Hello World!
```

Congratulations – you have successfully run your first program. Let's now understand the program we wrote above. The first line in the program above is a comment, indicated by the beginning **//**. In the next line, we declare a package for our program. Every program in Go must declare a package to which it belongs. The package **main** is special – it tells the Go compiler that this is an executable program. The other kind of program we can write in Go is a package, or in other words, a library which we use from another executable program. The next line (**import "fmt"**) imports the **fmt** package from Go's standard library. It implements the formatted Input/Output functions.

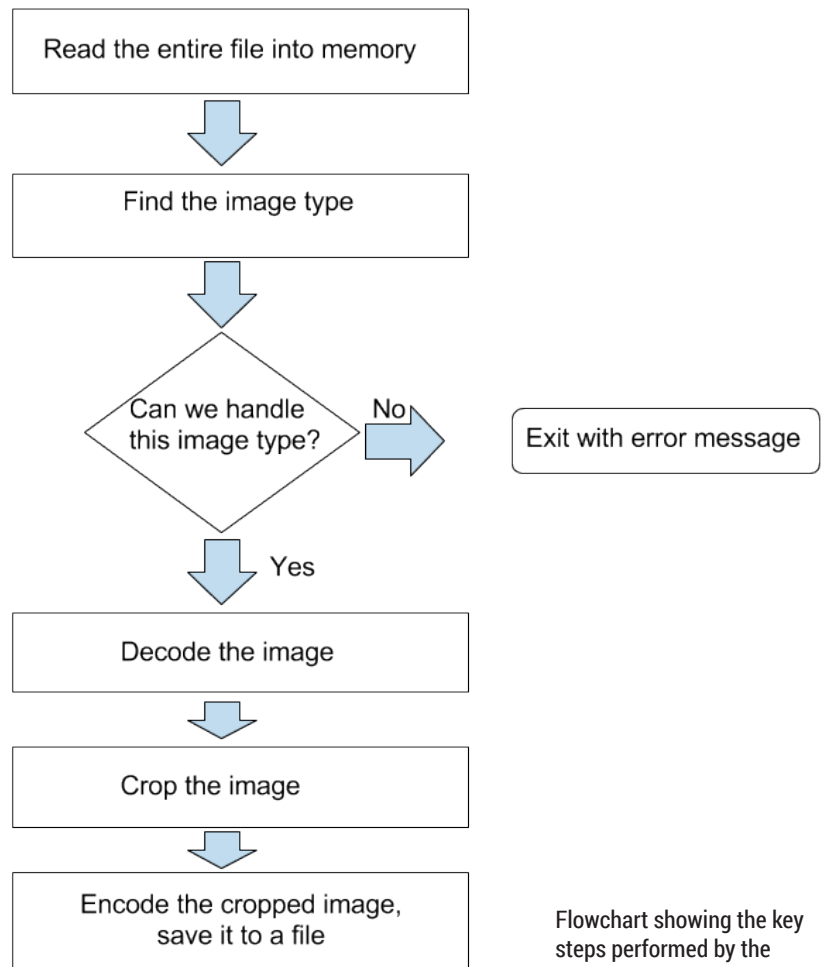
Next, we write our **main()** function – this is where the execution begins when you run a program in Go. Functions in Golang are created using the **func** keyword, followed by the function name and list of arguments it accepts in parenthesis, and then the return values. The **main()** function is special, as it neither accepts any arguments nor does it return anything. Next, we start the function body with an opening curly brace and have a single statement in the body – **fmt.Println("Hello World")**. This line calls the **Println()** function in the **fmt** package passing the string "Hello World!" to it.

The **Println()** function just prints the string followed by a new line.

The closing brace ends the function body.

We will now rewrite the program so that we can change what message we print by having the message stored in a variable:

```
// This is a simple program which prints a string
package main
import "fmt"
func main() {
    var shortMessage string = "Hello World"
    fmt.Println(shortMessage)
}
```



Flowchart showing the key steps performed by the **cropper()** function

```

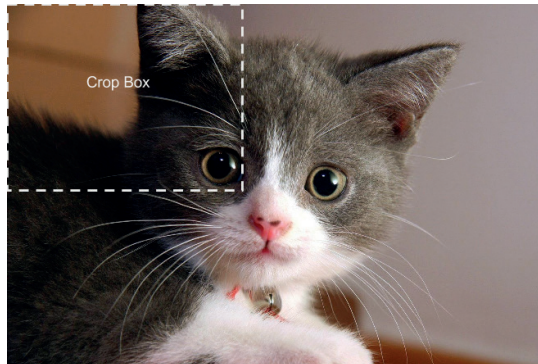
}
Every variable declaration in Golang must specify a type, indicating the data type of the values that we will store in it. The statement, var shortMessage string declares a variable, shortMessage, which will store a string. Here we assign the string "Hello World" to this variable. We then call the fmt.Println() function with this variable. Go supports a short-hand form of variable declaration inside functions using the := operator. Using that our above variable declaration would become shortMessage := "Hello World". This is however only valid when we also specify the initial value of the variable.
```

Besides the built-in data types such as integers and floating point numbers, programs can define a new data type using **struct**. For example, let's say that in our image cropping program, we want to define a data type that will store the image path, desired height and width that we want the image to be cropped into. We would create a new data type as follows:

```
type Config struct{
    path string
    height int
    width int
}
```

PRO TIP

Using **go run**, we can combine the building and running steps into one. For example, we can build and run **fact.go** as **go run fact.go** and you should see the output as above.



The passed-in desired height and width creates a "crop box" on the image starting at the top-left position.

The above code creates a new data type – **Config** – which has three fields: **path**, which is a string, **height** and **width**, both of type integers. We then create a variable of this type as follows:

```
image := Config{path:"/var/cat1.jpg", height:10, width:10}
```

Note that we used the **:=** operator here as well to create a variable of **Config** type. We can refer to each of the fields with the **.** (dot) operator. That is **image.path**, and so on. In our image cropping program, we will not be defining any new new type, but we will be using variables of struct type.

Writing your own functions

Except for the very rare cases, our program will usually be composed of functions other than the **main()** function. The final program we write by the end of this article is a more practical example, but here's a simple demonstration. The program below calculates the factorial of an integer (10):

```
package main
import "fmt"
func factorial(num int) int {
    fact := 1
    for i := 1; i <= num; i++ {
        fact = fact * i
    }
    return fact
}
func main() {
    num := 10
    fact := factorial(num)
    fmt.Printf("Factorial of %d is %d\n", num, fact)
}
```

We start by declaring the package and importing the **fmt** package. The function **factorial()** accepts an integer parameter and returns an integer. It starts by creating a variable, **fact**, and initialising it to 1. Then we have a **for** loop. If you are familiar with C, C++ or Java, this should seem familiar, but without the parentheses. The **loop** variable, **i**, is initialised with a value of 1 and continues until its value is greater than the number, incrementing by 1 in every step. The **++** operator in Go is used to increment the value in **i** by 1 in place. In the body of the **for** loop, we have the statement that actually calculates the factorial. Finally

we return the value in **fact** using the **return** statement.

The program execution begins at **main()**, where we declare a variable **num** initialised with 10, then call the **factorial()** function with this variable as an argument. The return value is stored in **fact**. Then, we print the factorial using the **Printf()** function in the **fmt** package. The **Printf()** function is used here because we want to substitute the values of variables **num** and **fact** string. Unlike **Println()**, **Printf()** doesn't automatically add a new line at the end of the output, so we have to do it ourselves.

Create a sub-directory **fact** in the sub-directory **src/github.com/linuxvoice** and type in the above program into a file **fact.go**. The **linuxvoice** sub-directory structure should now look like as follows:

```
linuxvoice/
├── fact
│   └── fact.go
├── helloworld
│   └── hello.go
```

Let's build and run the program from the **fact** sub-directory:

```
$ go build fact.go
$ ./fact
Factorial of 10 is 3628800
```

Writing the image cropping program

Now that we have seen the absolute basics, we'll get on to the image-cropping program. You can find the entire source in the file, **crop.go** in the repository https://github.com/amitsaha/linux_voice_1. Since this is a relatively long program, I will present and discuss the source code referring to snippets of the program from top to bottom.

The program starts with a block comment:

```
/* Simple command line program to crop images
Usage:
$ go run crop.go --height=5000 --width=7000 <path to>/
cat1.jpg <path to>cat2.png
The cropped images will be placed in the same directory
as the
original images with the file names being
cropped_<original_file_name>.<original_extension>
*/
```

When you have comments over multiple lines, you can have them as a block comment instead of prefixing each line with a **//** as we saw in our first program. The **/*** indicates the beginning of a block comment and ***/** indicates the end.

Similar to all our programs so far, our image-cropping program will be an executable program, so we specify the package to be **main** and then import a number of packages from the standard library as well as the third-party package **github.com/oliamb/cutter**. When you try to import a package that's not in the standard library, Go will look for it in the **src/ sub-directory** we created in the beginning. Thus, in this case, the **cutter** package should exist in **src/github.com/oliamb/cutter**. We will see how we can do so before we run our program.

```

package main
import (
    "bufio"
    "bytes"
    "flag"
    "fmt"
    "github.com/oliamb/cutter"
    "image"
    "image/jpeg"
    "image/png"
    "io/ioutil"
    "log"
    "net/http"
    "os"
    "path/filepath"
)

```

Next, we have the first function in the program:

```

func validateImageType(imageType string) bool {
    recognizedImageTypes := []string{"image/
jpeg", "image/png"}
    for _, t := range recognizedImageTypes {
        if imageType == t {
            return true
        }
    }
    return false
}

```

The `validateImageType()` function accepts a string argument, `imageType`, and returns a Boolean, specified by the `bool` return type. In the first statement, we create a slice of strings `recognizedImageTypes` consisting of the two image types that our program will be able to crop. A slice in Go is a data structure used to store a collection of elements of the same type. The position of an element in the slice is referred to as the index and starts at 0. It is similar to an array in C. Go also has arrays, but we shall not discuss them in this article.

Then we go over each element of the `recognizedImageTypes` array using a `for` loop and the `range` keyword. The `range` keyword returns the index and the element itself for each of the elements in the slice. Here is a demonstration of how `range` works in this case:

```

// Example of how range works
package main
import "fmt"
func main() {
    recognizedImageTypes := []string{"image/
jpeg", "image/png"}
    for i, t := range recognizedImageTypes {
        fmt.Println(i)
        fmt.Println(t)
    }
}

```

The above code will print:

```

0
image/jpeg
1
image/png

```

```
$ go run crop.go --height=500 --width=700 test_images/cat1.jpg test_images/cat2.jpg
```

Flags

Options

Flags and options recognised by our program

Since we don't care about the index in our final program, we use the throwaway variable, `_` (underscore) and use the variable `t` for the actual element. In the body of the loop, we check if the incoming `imageType` is one of the recognised image types using the `if` statement. If a match is found, we return true. If we haven't found a match after checking both the elements of the slice, we return false.

The next function in the program is the `cropper()` function, which performs the core functionality of the program. It accepts three parameters: a string containing the path to a image file and two integers corresponding to the desired height and width. The function then performs the steps outlined in Figure 2. Steps 1,2 and 3 are carried out by the following lines:

```

imageData, err := ioutil.ReadFile(inputFilePath)
if err != nil {
    log.Fatal(err)
}
imageType := http.DetectContentType(imageData)
if ! validateImageType(imageType) {
    log.Fatal("Cannot handle image of this type")
}

```

The first statement uses the `ReadFile()` function from the `ioutil` package to read the entire file into memory. The function returns two values: the data as bytes and an error value, which are stored in the variables `imageData` and `err` respectively. If the value in `err` is not equal to `nil` (a special value indicating an absence of a valid value) or in other words, if there was an error when reading the file, we call the `Fatal()` function from the `log` package passing the error. This will print an error message and terminate the program execution.

PRO TIP

The `Fatal()` function essentially does two things – it prints the message associated with the error – when it is called with a error value, else the string passed to it, and calls the `Exit()` function from the `os` package, thus terminating the program.

Next, we detect the image type by using the `DetectContentType()` function in the `http` package.

We call it passing the bytes we just read into `imageData` as an argument. It returns a string corresponding to the image type of the supplied image. Once we have the image type, we call the `validateImageType()` function we wrote earlier. If the function returns false, we call the `Fatal()` function from the `log` package with a message, and our program will exit after printing the message.

If, however, the image is one of the recognised image types, we proceed with Step 4, which is implemented by the following code:

```

reader := bytes.NewReader(imageData)
img, _, err := image.Decode(reader)

```

```
if err != nil {
    log.Fatal(err)
}
```

The **Decode()** function in the **image** package is used to decode the bytes in **imageData**. However, the **Decode()** function expects the data as a **Reader**, and hence we use the **NewReader()** function of the **bytes** package to create a new **Reader** that will read from the bytes in **imageData** and then pass it to the **Decode()** function. This function returns the decoded image of type **Image**, the format of the image as a string and an error value. Since we don't care about the format of the image any more (we already know it), we have used the underscore instead of using a variable. If the value of **err** is not equal to **nil**, we call the **Fatal()** function passing **err** to it, else we proceed to the next step.

The following code implements Step 5 above:

Our program will support cropping multiple images at once and expects the paths to the images as arguments

```
croppedImg, err := cutter.Crop(img, cutter.Config{
    Height: cHeight,
    Width: cWidth,
})
if err != nil {
    log.Fatal(err)
}
```

We call the **Crop()** function from the **cutter** package passing the decoded image and a structure variable of type **Config** (defined in the **cutter** package), indicating the desired height and width (Figure 3). This function returns the cropped image and an error value. Once again, we check if the **err** value returned is not **nil** and if so, we call **Fatal()** passing the **err** value so that it prints the error and exits.

Before we can implement Step 6, we have to create a new file where we will store the cropped image:

```
croppedFileDir, fileName := filepath.Split(inputFilePath)
```



Original image (Source: <http://scienceblogs.com/gregladen/files/2012/12/Beautiful-cat-cats-14749885-1600-1200.jpg>)

```
croppedFileName := fmt.Sprintf("%scropped_%s",
croppedFileDir, fileName)
```

```
croppedFile, err := os.Create(croppedFileName)
```

```
if err != nil {
    log.Fatal(err)
}
```

```
defer croppedFile.Close()
```

The cropped image will be stored in the same directory as the original image into a file **cropped_<original_filename>.<original_extension>**. That is, if your original file was in **/var/images/cat1.jpg**, the cropped image will be saved in **/var/images/** as **cropped_cat1.jpg**. Hence, we will need to find the directory of the file supplied as well as the filename (and extension). We can get both using the **Split()** function from the **path/filepath** package. It returns two values: the directory and the filename, which we store in **croppedFileDir** and **fileName** respectively. The directory name is returned with a trailing **/**. Now that we have the directory and the filename of the original image, next we create a string, **croppedFileName**, using the **Sprintf()** function from the **fmt** package, whose value will be the location of the cropped file. Then, we call the **Create()** function in the **os** package to create the file passing it the value in the variable **croppedFileName**. It returns two values: the first referring to the opened file, and the second an error value. We check for the value of **err** to see if there was any error while s the file and call **Fatal()** otherwise. The next statement uses the **defer** statement to specify that just before the function returns, it should close this file using the **Close()** function.

Finally, we can implement Step 6 – encode the cropped image and write it to the above file:

```
croppedFileWriter := bufio.NewWriter(croppedFile)
```

```
if imageType == "image/png" {
```

```
    err = png.Encode(croppedFileWriter,
croppedImg)
```

```
}
```

```
if imageType == "image/jpeg" {
```

```
    err = jpeg.Encode(croppedFileWriter,
croppedImg, &jpeg.Options{100})
```

```
}
```

```
if err != nil {
    log.Fatal(err)
}
```

```
croppedFileWriter.Flush()
```

The first statement creates a new **Writer** to write to **croppedFile**. This is required for the encoding operation. The encoding function we use will depend on the image type, and hence we use an **if** statement to check the image type and call the appropriate encoding function. If the image type is **image/png**, we call the **Encode()** function in the **image/png** package with two arguments: the writer where we want to write the cropped image to and the cropped image that we want to encode. Similarly, if the image type is **"image/jpeg"**, we call the **Encode()** function in the **"image/jpeg"** package. The first two arguments to the

function are the same in this case, and there is an additional argument – a variable of the `jpeg.Options` struct type indicating the quality of the encoding we desire. Here we specify the quality as 100 (the highest quality).

Both the functions return an error value. Next, we check if there was an error, calling `Fatal` if there was one, else we call the `Flush()` function so that all the data is written to disk.

The final function in our program is the `main()` function. It does the following things:

- 1 Set up the flags, which enables us to specify the input as command line flags and arguments.
- 2 Check if all the required input has been supplied to the program.
- 3 Call the `cropper()` function above with each of the images specified and the desired crop height and width.

Step 1 above is implemented by the following code:

```
cHeight := flag.Int("height", 0, "Crop Height")
cWidth := flag.Int("width", 0, "Crop Width")
flag.Parse()
```

The `flag` package is used to specify command line flags for our program. We expect both the height and width to be specified as integers. Hence, we call the `Int()` function twice to set up the flags for height and width storing the returned value in `cHeight` and `cWidth` respectively. The first argument to the function is the flag that will be specified, the second argument is the default value if not specified, and the third is a short description of what the flag means. The third line above calls the `Parse()` function, which essentially looks at the command line arguments supplied to the program when run and stores the value of the flag in the correct variables above.

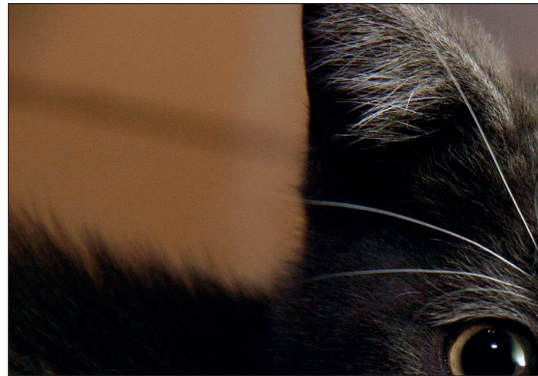
Next, we implement Step 2. The following code snippet checks if both the flags were supplied and a non-zero integer specified:

```
if *cHeight <= 0 {
    log.Fatal("Must specify the crop height to be
    positive integer")
}
if *cWidth <= 0 {
    log.Fatal("Must specify the crop width to be a
    positive integer")
}
```

Note that we have used `*cHeight` and `*cWidth` to refer to the values in the two variables instead of just `cHeight` and `cWidth`. This is because the `Int()` function above returns the address of the values. If the value in any of the variables is not a positive integer, we terminate the program.

Our program will support cropping multiple images at once and expects the paths to the images as command line arguments. So we need to check if at least one was supplied. The next snippet does that:

```
numImages := len(flag.Args())
if numImages == 0 {
    log.Fatal("Must specify at least 1 image to crop")
}
```



Cropped image of height and width set to 500 and 700 pixels respectively.

The `Args()` function returns all the command line arguments that were not flags as a slice. We use the `len()` function to find the number of arguments specified, and if it is 0 we terminate the program specifying an error message.

The following code snippet implements Step 3:

```
for _, inputFilePath := range flag.Args() {
    cropper(inputFilePath, *cWidth, *cHeight)
}
```

As earlier, we use a `for` loop and `range` to go over each of the image paths supplied as command line arguments. In the body of the loop, we call the `cropper()` function with the input file path and the cropping width and height.

Cropping images

Now that we have gone through the entire program, it's time to get the code and run it. From your terminal, run the following command:

```
$ go get github.com/amitsaha/linux_voice_1
```

You should now see two sub-directories under your `src/github.com` sub-directory, `amitsaha` and `oliamb`. Under `amitsaha`, you will see a `linux_voice_1` package which has our program, `crop.go`, along with a few other files. Under `oliamb`, you will find the `cutter` package, which is the package we use to crop the images. What we saw here is that `go get` was able to find out that our code in `crop.go` used a third-party package and it downloaded that as well and placed it in the right location. This is why we need `GOPATH` to be set (among other things). Now, `cd` into the `github.com/amitsaha/linux_voice_1` and run the program:

```
$ go run crop.go --height=500 --width=700 test_images/
cat1.jpg
```

The original image (Figure 5) will be cropped and the cropped image will be placed in the `test_images/` sub-directory as `cropped_cat1.png` (Figure 6).

We started with the complete basics of Go in this article, and quickly went on to writing a non-trivial command line program to crop images. Along the way, we learned about declaring variables, defining data types, writing functions, handling errors and working with several Go packages. 📖

Amit Saha is the author of *Doing Math with Python* (No Starch Press) and a software engineer. He blogs at <https://echorand.me>, and can be reached at amitsaha.in@gmail.com

THE FUTURE OF COMPUTER LANGUAGES

The future is unknowable... but when it comes to coding we can have a good guess.

JULIET KEMP

WHY DO THIS?

- Speak expansively on the future of programming in your next job interview.
- Inform your choice of your next language to learn.
- Learn from the past.

After looking at a few of the many computer languages that have been invented over the last 50-odd years, it's time to polish up that crystal ball and attempt to make some predictions about the next 50 years... or at least the next 20. Are the older languages that are still in use today coming to the end of their useful life? What new languages are coming up that are going to take the world by storm? What are the current big problems that modern programmers and language designers are trying to solve? And how is the practice of programming going to change in the future?

Looking at languages over the last 30–50 years, it's apparent that there are a bunch of 'old' languages that are still firmly in use. Fortran, COBOL, C, Ada, and others are still out there and aren't in danger of disappearing any time soon, though some of them are perhaps a little more precarious than others.

Some older languages are in use primarily for legacy systems (especially old mainframe software) and it's possible that those systems might gradually be ported over to newer languages. Since these languages don't see much new development, if that happened they would gradually become obsolete. More likely however is that newer hardware (newer

mainframes, virtual machines, and cloud computing) will be used to replicate the old hardware and run the same code; that's actually easier, cheaper, and safer (from a project management point of view) to do than rewriting the code.

C, on the other hand, is actively used in modern development and still underpins large swathes of software. The Linux kernel is, of course, written in C. C is outstandingly good at what it does, and it's hard to imagine its dominance for close-to-the-metal development disappearing any time soon.

So would it be surprising if Fortran, C, and COBOL code is still chugging along happily in particular niches in another 40 years? Not even slightly.

The middle-aged languages like Java, JavaScript, and Python are all still under active development, and in very heavy use. The way that people are working in these languages has changed (see section below); but the languages themselves are well entrenched and will more than likely still be around in 20 years time. On the other hand, 15 years ago Perl was everywhere, whereas now its profile is much lower; however, there's still lots of Perl out there and it remains a useful language. (And, of course, Perl 6 will surely arrive at some point and may change the scripting landscape again.)

Older functional languages like Scala (2001) and Haskell (1990) have recently risen in popularity. Facebook and Google both use Haskell for some of their work; and LinkedIn, Twitter, and Tumblr all use Scala. This reflects a generally increasing interest in functional programming, with functional ideas and features appearing in other mainstream languages (such as Java 8's introduction of lambdas). Whether this is the start of functional programming's forthcoming reign of glory (as functional programming enthusiasts argue), or whether it's just a brief flurry of functional ideas being pinched by other languages, is as yet unclear. Perhaps most likely is that as more people get to grips with functional programming, it will become more popular for those problems that it solves particularly well, but that other languages and paradigms will continue to be popular in their own niches. More generally, the trend suggests that the same thing may happen to other languages at various times, as their particular features become



Are mainframes, and the software that runs on them, on their way out?

suddenly relevant to a modern problem (see the concurrency section below).

Another area of development hooks new (or new-ish) things into existing, older, languages. More and more languages are being written to run on the Java Virtual Machine – Clojure, Groovy, JRuby and Jython are the best-known, but there are a couple of dozen others. Does this suggest a general move towards a single platform that can be coded in multiple languages? It's an idea with obvious practical advantages if it works well; packaging and installation become potentially easier, while programmers can work with the higher-level language that best suits them and their problem domain. It does mean carrying the overload of an extra virtual machine, but increasingly virtual machines (of whatever sort) are part and parcel of how software is run.

Concurrency

Concurrency is possibly the biggest issue facing modern programmers. It exists when multiple computations are executing at the same time, and may be interacting with one another. This can be on cores within a single chip, in threads on a processor, or on multiple processors/machines. Because the computations aren't just parallel, but may interact (and share resources), there are numerous issues to tackle for the process to be successful, including data exchange, memory allocation, and scheduling. Concurrency is increasingly a big deal as it becomes more common (and cheaper!) to run multiple threads/cores/processors at the same time.

Various potential solutions or resources to support solutions are available and/or under experimentation. Erlang (which dates back to 1986) has grown in popularity recently, being used for cloud-based high-performance computing that needs concurrency. Erlang has concurrency 'baked in', with a set of primitives that create processes and handle inter-process communication; these processes form the basic structure of an Erlang program. Erlang also avoids the problem of mutable shared data and lock access by only passing messages and not having

```

hello.erl (*) - GVIM
File Edit Tools Syntax Buffers Window Help
-module(hello).
-export([hello_world/0]).

hello_world() -> io:fwrite("Hello world!\n")

julieta@debian: ~
File Edit View Search Terminal Help
julieta@debian:~$ erl
Erlang/OTP 17 [erts-6.2] [source] [async-threads:10] [kernel]
Eshell V6.2 (abort with ^G)
1> c(hello).
{ok,hello}
2> hello:hello_world().
Hello world!
ok
3>
"hello.erl" [New] 4L, 89C w

```

Hello World in Erlang. Save the file, then execute it in the Erlang shell.

any mutable shared data at all. This means that each process has its own private state, and processes change state based on their own behaviour or on explicit messages from other processes, not on a shared structure that any process can access. (Shared structures have the problem that they must be locked while being accessed; if a process dies while it has a lock, this can cause issues for other

The middle-aged languages like Java, JavaScript and Python are still under active development

processes trying to access the same structure.) Erlang is also functional, and its focus is scalability, high availability, and no downtime. This reflects its telecom industry origins; telecoms demand absolutely minimal downtime. This makes it a good match for modern web and communications services (WhatsApp uses Erlang). The Erlang VM handles all the scheduling and load-balancing, meaning that the programmer doesn't have to.

Another approach to parallelisation is the language CUDA (or the open-source OpenCL), which utilises the

A, B, C... D?

C and C++ have been around for a long time and are hugely successful, as discussed in the previous section. D is intended as a re-engineering of C++, keeping its good parts and in particular its performance, while making it more convenient for programmers, taking inspiration from the lessons learnt from practical programming experience in C++. It ditches some C++ features, and adds a bunch of extra ones, including unit tests, garbage collection, and various array features. It also adds an inline assembler; illustrating that, like C/C++, D is intended to allow the programmer to access low-level processor features as required. Its creators claim that D is suitable for everything, from high-performance back-end code to quick-and-dirty scripting.

```

void main() {
    writeln("Hello world!");
}

julieta@debian: ~
File Edit View Search Terminal Help
julieta@debian:~/coding/future$ gdc hello.d
hello.d:4: error: unterminated string constant starting at hello.d:4
hello.d:5: error: found 'EOF' when expecting ','
hello.d:6: error: found 'EOF' when expecting ';'
hello.d:7: error: found 'EOF' when expecting ':' following statement
hello.d:8: error: found 'EOF' when expecting '}' following compound statement
julieta@debian:~/coding/future$ gdc hello.d
julieta@debian:~/coding/future$ ls
a.out hello.d
julieta@debian:~/coding/future$ ./a.out
Hello world!
julieta@debian:~/coding/future$ gdc hello.d -o hello
julieta@debian:~/coding/future$ ./hello
Hello world!
julieta@debian:~/coding/future$

```

Compiling D with the `gdc` compiler. (All those error messages are just one missing semi-colon.) Note the need to specify output file with `-o`.



This is a Swift playground on OS X, with code on the left, results on the right, and a pop-up box showing the result of a particular line. Playgrounds aren't (yet?) available on Linux, though the language as a whole can be downloaded from swift.org.

processors in modern graphics cards (GPUs) when they're not in use for graphics, thus running in parallel. This works really well for certain problems, and is popular with academics. Effectively, CUDA translates the data into graphical form and then uses the GPU to 'look' at and analyse it, as GPUs are optimised for image manipulation.

Perl 6 and Clojure are also being promoted as suitable languages for solving concurrency problems. Clojure (a variation on Lisp) is intended to make it "simpler to design and implement parallel algorithms", and compiles down to the JVM. Perl 6 introduces easy (or easier) concurrency in a dynamic programming language. However, it's anyone's guess when Perl 6 (or rather, a complete implementation of the Perl 6 standard) is actually going to be released.

Google's new language Go (unveiled in 2009) is designed to handle concurrency well, although the major aim was for Go to be hugely clean and "simple enough to hold in one programmer's head". Go is compiled, statically typed, and aimed at systems programming. (As of this year, it's also available for mobile device and smartphones, another nod to the importance of mobile-first coding.) Go code is intended to be concise and readable, and, rather like Swift (see below), it uses interfaces and type embedding over virtual or non-virtual inheritance. Flexibility, fast development, and a certain kind of simplicity are important. See p84 for our tutorial!

Web-first and mobile-first

The other big current coding issue is web-based and mobile-based coding. More and more software is run in browsers (the ultimate example being ChromeOS), which makes JavaScript and other web languages increasingly important.

Currently JSON is the data-object format of choice, with libraries in every major language, and web services are RESTful; but 10 years ago it was all about XML and SOAP. So in another 10 years will we look back from a similar distance on JSON and REST? Web-coding is still moving pretty fast, so this is one of the most likely areas for things to change noticeably in the reasonably near future.

The importance of mobile-first coding suggests that Java will be around for a while, as Android (a version of Java) has the majority of the smartphone and tablet market. The other big chunk of that market belongs, of course, to Apple and to iOS; making Swift, Apple's new iOS/OSX/watchOS language, which replaces Objective C, one of the most important languages to watch.

Swift was released in mid-2014, is growing very fast and is now open source. According to Apple, it's all about "protocol-oriented programming". Protocols, in Swift, are rather like interfaces; and in Swift, instead of creating classes and subclasses, you are encouraged to create protocols and then create types and structs that use them. Protocols can also be extended with default behaviours. This makes Swift highly flexible; in particular because a type can have multiple protocols very easily, whereas multiple class inheritance is not possible in all languages and can be top-heavy where it is possible. Protocols can also be used by more than just class types. It's also easy to extend existing protocols using the extension keyword, at which point anything that conforms to this protocol now also conforms to your extension.

Conveniently, Swift co-exists with Objective C in the same app, so developers can move over to it easily. It also includes a lot of interactive development technology, in particular the "playground" section of the IDE, which enables the developer to try out pieces of code. (This is also explicitly aimed at helping people learn to code.) And there's a strong incentive for developers to move over to Swift, because there's such a huge market for iOS/OS X software, and it has such a low barrier to entry. If you're starting out in the Apple universe, you'd choose Swift over Objective-C, and even if you're already an Obj-C coder you might well jump ship.

Swift shows off a few modern coding ideas – high preference for flexibility and speed of development, and a fondness for interactive development

Big Data

The growing importance of big data means that R (a statistical language, around since 1997) is becoming much more popular, and is used by companies such as Google, Facebook, and the New York Times. However, it's suggested that it can be slow with big data sets and is better used for prototyping than for building models. Matlab (a numerical programming language) has always been heavily used in the scientific community but opinion is divided on whether it is on the way out or not. The big recent improvements in Python's maths and statistical libraries may make Python a more appealing choice for numerical programming. IPython and NumPy (designed to make Python competitive with Matlab for numerical computing) are good for scratchpad use, and Python's data community has many toolkits available. Matlab definitely still has its niches though, despite being an expensive piece of non-free software. Both Matlab and R have parallelisation packages available.

Julia is similar to Python, but runs much faster. It's created specifically with scientific number crunching in mind, with powerful concurrency and networked programming facilities, and interfaces available with Fortran and C library routines. With the week-long runtimes of some scientific computing, the increase in speed that Julia provides is important (taking 1 second rather than 4 isn't a big deal; taking a week rather than four weeks definitely is). Julia is super-fast, potentially more scaleable than Python, and easy to learn; but the community is in the early stages, and currently it just doesn't have the libraries and toolkits to compete with Python or R.

The Internet of Things

The Internet of Things is the newest buzz-phrase and no one quite knows where that's going. Some applications are straightforwardly nonsense (the classic 'smart fridge', for example), but the 'internet of things' covers a lot of ground, and other parts of it do make more sense. This certainly has the feel of something that really could change dramatically in the foreseeable future, and if that happens, then changes in coding languages, and perhaps in coding practice, are very likely to follow. Might 3D printing evolve to the point where we need compilers that compile actual objects, rather than code? What would that even look like?

environments and for code automation (the compiler will make suggestions for you to improve your code).

Coding: learning and practice

There's a lot more available for younger coders than there used to be. Raspberry Pi and Scratch are both supporting different ways of learning to code, and learning for quite young children (especially Scratch). Robots Dash and Dot also aim to introduce the ideas behind coding to young kids, and then support them in learning slightly more complicated ideas and algorithms. The focus is on solving problems in a coding-type way. A new generation of coders coming up from this background might tackle programming and languages in a brand new way.

Coding jobs are also starting to look different. More coding in the cloud and easier code-sharing means that teams might be spread across the world and never or rarely meet one another. Cloud deployment means that you might not know the sysadmins responsible for deploying and managing the software you write in the same way that you might have done at one time. Then again, there's more scope now for writing software that is accessed from your own servers, rather than sending out disks to be installed on a client's systems.

Various forms of increased automation give programmers more time to focus on the important stuff. Libraries, frameworks, APIs and plugins mean that programming can now be more about sticking existing things together than about writing entirely new things – but that also frees you up to focus on the global application logic, and widely-used libraries and frameworks make for more exhaustive user-testing and thus fewer bugs. Python's recent growth is partly a reflection of the expansion of its libraries, making it easier to stick scripts together quickly. A problem can be if multiple competing frameworks exist; knowing which to use may be a challenge, and it reduces some of the advantages of having shared frameworks in that the user base for each framework will be smaller.

IDEs and other code extensions can also generate more code for you, especially in boilerplate-heavy languages. An example is JavaScript, where updates like CoffeeScript and Less.js are making it easier to write JavaScript and complicated CSS.

Automated testing, test-first coding, push-on-green, and other similar practices can also make for tighter, more reliable code, if used properly, and thus for less time spent fixing bugs after software deployment. Improvements in version control systems, commit hooks to fire off automated tests, and so on, certainly mean that a programming job looks different now from how it did 10 years ago (and still more compared to 30 years ago).

Farther Future

Is Moore's Law coming to a slowdown? Does this suggest that something else will take its place, the same way that integrated circuits replaced transistors (which in their turn had replaced relays and vacuum tubes)? Ray Kurzweil argues that this will happen, and tips nanotubes for the most likely replacement. (He also argues that computational capacity will continue to grow exponentially, which leads to a bunch of stuff about the Singularity that I am not going to get into.) Reconfigurable hardware is another idea being thrown around, and is starting to be implemented in limited ways. New hardware often drives new software, and without knowing what exactly what that will look like (if it happens), it's nearly impossible to make predictions about it.

Looking at the last 40 years, however, hardware has changed a fair amount, and software has changed along with it, but we are still using some of the same

What's fascinating is the way that new purposes and new problems will arise, and how programmers will solve them

languages that were popular then. By extrapolation, therefore, we'll still be using some of the languages we're using now in another 40 years' time. Coding practices have changed, though, and the languages have often changed along with them. Modern Fortran looks a fair bit different to original Fortran; Java 8 is a long way away from Java 1.

The fundamental problem is that no single language can possibly have every feature any programmer could ever want, because some of them are incompatible with one another. So any vision of the One True Programming Language is doomed from the start. (Though many languages compiling onto a single virtual machine that then runs on many other machines is more feasible.) We'll always have multiple languages for multiple purposes, and we'll always have multiple programming paradigms for multiple purposes. What's fascinating is the way that new purposes and new problems will arise, and how new programmers will find new ways to solve them, with languages both new and old. 📺

Juliet Kemp is fluent in over 6,000,000 forms of communication, including esoteric languages such as Perl.



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.

Asynchronous Disk I/O

Spawn tons of disk I/O requests, and do something else while they are in flight. Just like with sockets, albeit without `epoll()`.

Back in LV016, we dealt with asynchronous networking I/O. This way, network services serve one request while waiting for a database to supply data for another. Asynchronous I/O greatly improves the service's capacity. No wonder it is a workhorse behind many popular web frameworks, like JavaScript's Node.js or Python's Tornado.

But network services rarely exist in the void. The data they serve ultimately comes from some storage, often a file. If access to this file is blocked, it stops the entire event loop. This hurts performance and is something we'd rather avoid. Is there a way to treat a file asynchronously, like a socket? Meet a rather obscure and often misunderstood Linux feature: asynchronous disk I/O!

Traditional file access is blocking. This doesn't scale very well, yet it makes code more straightforward:

```
char buf[1024];
int fd = open("somefile.txt", O_RDONLY);
```

```
read(fd, buf, sizeof(buf));
```

Here, `read()` won't return until it puts some data in `buf`, or an error occurs. Sequential execution is easy to reason about. The downside is this code can't do anything useful while it waits for `read()` to complete, and it can take time on a busy disk.

One can also set `fd` to non-blocking mode:

```
int fd = open("somefile.txt", O_RDONLY | O_NONBLOCK);
```

Or, via `fcntl()` system call:

```
fcntl(fd, F_SETFL, O_NONBLOCK);
```

In non-blocking mode, if `read()` can't complete immediately, it returns `(-1)` and sets `errno` to `EWOULDBLOCK`, or `EAGAIN` if `fd` is a socket. It is quite common to check for both:

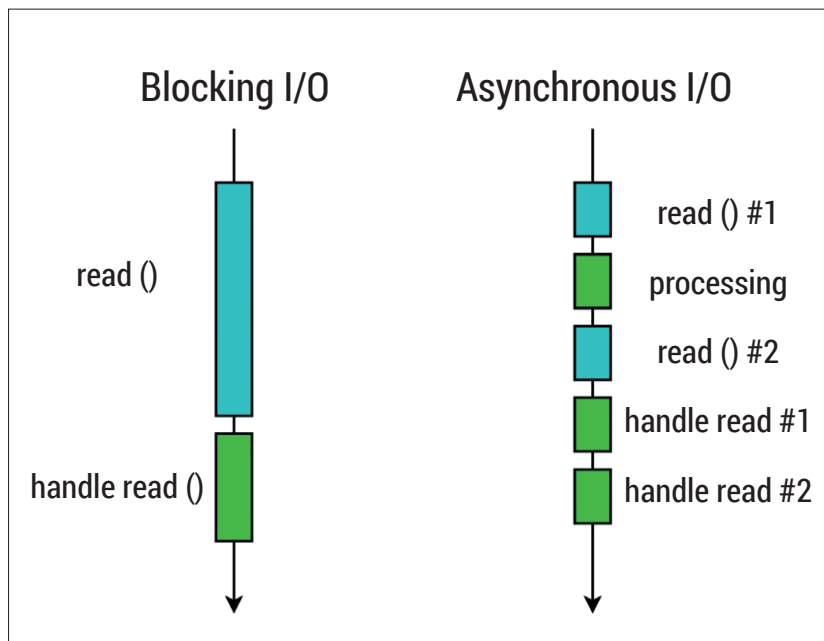
```
res = read(fd, buf, sizeof(buf));
if (res < 0 &&
    (errno == EAGAIN || errno == EWOULDBLOCK)) {
    /* handle gracefully */
}
```

Of course, you can busy yourself waiting for the descriptor until `read()` succeeds, but it would be rather inefficient. You may think about feeding `fd` to `epoll()` or similar (LV016) – it worked for sockets, didn't it? Bad news everyone: this time, it won't. With `epoll()`, you'll simply get an `EPERM` error. `poll()` and `select()` will run, but immediately report `fd` readable regardless of whether the read would block or not.

Using threads

Perhaps a more traditional (and more popular) way to non-blocking file I/O is via threads. Threads are worth a Core Tech of their own (please drop us a line if you are interested in this subject), but in a nutshell the idea is simple. When you want to read data from a file, you spawn a new thread. This thread issues a blocking `read()` and waits for it to complete, while the event loop continues normally in the original thread. When `read()` returns, the main thread gets some notification. Threads execute in parallel, so this is much like the behaviour we are after. No wonder many applications

Blocking vs Asynchronous I/O time diagram. The latter interleaves I/O and processing, resulting in better capacity.



and frameworks take this route. Say, Node.js (or, more precisely, **libuv**) behaves exactly this way.

Threads have their own issues and arguably make your code harder to debug, but it's only part of the story. The real problem is that threads are a rather costly abstraction to create and maintain. The latter is usually mitigated with so called "thread pools": you create a number of threads (say, 32) in advance and let them idle. To defer something to a thread, one posts a message to shared queue to which all threads "listen". The first thread to "hear" the request handles it. If a thread remains idle for some time, the application may decide to kill it. Similarly, if the application detects that threads in the pool are mostly busy, it may spawn additional ones to increase the capacity.

POSIX AIO

Asynchronous file I/O (AIO) APIs in Linux come in two flavours. Both provide a way to issue (or enqueue) an I/O operation, to retrieve its status as it finishes, and to cancel the operation you are no longer interested in. Both have some notification mechanisms to tell your application the operation has completed (that's the main idea). The difference is in design, implementation, and (naturally) interfaces (function names, structures and so forth).

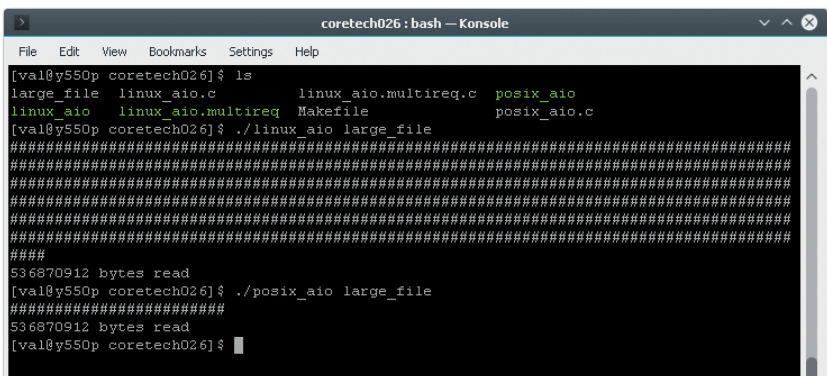
The first API we are going to cover is POSIX AIO. As the name suggests, it's part of the POSIX standard and is available across many Unixes (which is good). Note that POSIX dictates interfaces, not implementations: we'll revisit this point later. On Linux, the POSIX AIO comes in **librt** (a part of *Glibc*) with other fancy stuff like POSIX message queues and process timers. Most POSIX AIO calls have the **aiocb** prefix, and they are declared in **aiocb.h**.

POSIX AIO has two main notification means: a signal, and a threaded callback. Neither are particularly convenient for applications built around event loops, albeit **signalfd** may help you a bit (see boxout). Notifications are configured per I/O request. For a signal, you choose a number (usually **SIGUSRx**). For thread notifications, you supply a function which executes "as if" it were an entry point for some thread. Either case, you may choose a value to pass to the callback. It could be an arbitrary integer or **void ***. The latter is commonly used to send a pointer to some application-specific structure. It is much like the **this** pointer in an object-oriented languages like C++.

Let's dissect a simple POSIX AIO example. It reads some (presumably large) file and prints a hash mark simultaneously, to prove the operation is really asynchronous. You'll find the complete sources on the LV website.

```
int fd = open(argv[1], O_RDONLY);
```

First, we open a file. Note that ordinary blocking mode is used, as asynchronous I/O and non-blocking I/O are different things. Then, we prepare the request. POSIX AIO represents requests with **struct aiocb**, which is the same for reads, writes and sync operations. We allocate one statically and zero-fill it



(which is recommended) at the beginning of **main()**:

```
struct aiocb request = { 0 };
request.aiocb_fildes = fd;
request.aiocb_offset = 0;
request.aiocb_buf = buf;
request.aiocb_nbytes = LARGE_CHUNK;
request.aiocb_sigevent.sigev_notify = SIGEV_THREAD;
request.aiocb_sigevent.sigev_notify_function = &handle_read;
request.aiocb_sigevent.sigev_value.sival_ptr = &request;
```

Here, we set the descriptor (**aiocb_fildes**), the buffer to receive data (**aiocb_buf**), and the number of bytes to read (**aiocb_nbytes**), as in ordinary **read()**. **aiocb_offset** is file offset, as in **pread()**, but it's not optional here. Asynchronous requests can complete in any order, so "current file position" doesn't make sense. There are some additional fields you may want to fill; consult **aiocb(7)** for details.

```
request.aiocb_sigevent.sigev_notify = SIGEV_THREAD;
request.aiocb_sigevent.sigev_notify_function = &handle_read;
request.aiocb_sigevent.sigev_value.sival_ptr = &request;
```

Then comes the callback. Here, we opted for thread notify (**SIGEV_THREAD**), and send it a pointer to the request (**sival_ptr**). Generally you shouldn't let pointers to stack-allocated values leave the function, but for this simple example, it will work fine.

Finally, we kick off the operation with:

```
aiocb_read(&request);
static void handle_read(union sigval signal)
{
    struct aiocb *request = signal.sival_ptr;
    int res;
    res = aio_error(request);
    if (res < 0) {
        /* Request is still in progress or cancelled */
    } else if (res > 0) {
        printf("\nError: %s\n", strerror(res));
    } else {
        printf("\n%d bytes read\n", aio_return(request));
        /* Use request->aio_buf to read the data */
    }
    done = 1;
}
```

First, it checks if the request completed successfully. If this is not the case, **aio_error()** returns

Our samples print hash marks to show you they can do something while data loads from the disk.

```

2707 #include <libaio.h>
2708
2709 #define(10_event, 0x0, sizeof(10_event));
2710
2711 byte* buf = static_cast<byte*>(0_malloc(pages)(ENV_PAGE_SIZE * 3));
2712 byte* ptr = static_cast<byte*>(0_malloc(buf, ENV_PAGE_SIZE));
2713
2714 struct iocb iocb;
2715
2716 /* Suppress valgrind warning */
2717 memset(buf, 0x0, ENV_PAGE_SIZE * 3);
2718 #define(1iocb, 0x0, sizeof(iocb));
2719
2720 struct iocb* p_iocb = &iocb;
2721
2722 if (test_read_only_mode) {
2723     io_prep_pwrite(p_iocb, fd, ptr, ENV_PAGE_SIZE, 0);
2724 } else {
2725     ut_at(ENV_PAGE_SIZE >= 512);
2726     io_prep_pread(p_iocb, fd, ptr, 512, 0);
2727 }
2728
2729 int exe = io_submit(10_ctx, 1, p_iocb);
2730
2731
2732 if (exe >= 1) {
2733     /* Now collect the submitted IO request. */
2734     exe = io_getevents(10_ctx, 1, 1, aio_event, 0);
2735 }
2736
2737 ut_free(buf);
2738 close(fd);
2739
2740

```

MySQL (and its clones) implements Linux AIO support through the *libaio* wrapper.

a positive integer which is the same as **errno** for synchronous requests. Next, we print the number of bytes read. **aio_return()** yields what would be the return value of synchronous **read()** or **write()**. Finally, the callback sets the flag that tells **main()** to stop:

```

/* Simulate event loop */
while (!done) {
    fprintf(stdout, "#");
    fflush(stdout);
    usleep(LOOP_TIMEOUT_USEC);
}

```

Note that we ignored all possible threading issues for simplicity's sake. The **aio(7)** man page provides a much longer and more elaborate example. It uses signal notifications, and also shows how to cancel the request.

Frankly speaking, POSIX AIO is rather unpopular in Linux, and you'll hardly meet it in the wild. One exception is the *lighttpd* web server (**www.lighttpd.net**), which proudly introduced AIO support in version 1.5. However, *lighttpd 1.5* was a preview release which dated back to 2009 and never became stable.

One reason is that many applications don't really need asynchronous disk I/O. Database management systems or similar software that juggles gigabytes of data under heavy load can benefit from it. For the rest of us, the game is often not worth the candle. Another, and perhaps more compelling reason, is that Linux provides no kernel support for POSIX AIO. Under the hood, *Glibc* implements this API entirely in user space via threads, so all the usual limitations apply. Leave POSIX AIO for using in portable code (if at all); Linux has something else to offer.

Native AIO

Linux has introduced "native" AIO in version 2.6. This is true kernel space implementation, yet unsuitable as a backbone for POSIX AIO, as we'll learn shortly. Linux AIO is also missing from *Glibc*, leaving you with two options. First, you may use **syscall()** to run Arch-specific system calls directly by their numbers. Or, there is the **libaio** library (<https://git.fedorahosted.org/cgit/libaio.git>), which wraps everything for you. The wrapper is in fact rather thin, so many developers prefer the former. Our example will use **libaio** though, to hide some small yet unnecessary details.

Linux AIO is generic API, but it was likely designed with databases in mind. As such, it expects your code to meet some requirements, which may look strange unless you write a DBMS. You must open files with the **O_DIRECT** flag to disable caching at kernel level. **O_DIRECT** expects that userspace buffers, their sizes and file offsets are aligned on a 512-bytes boundary. All of this sounds like Linux AIO is better suited for accessing raw devices (say, **/dev/sda1**) than filesystem data. If you miss **O_DIRECT**, the code would still compile and run, but may eventually resort to blocking I/O. These requirements are not a problem for a database engine, which usually works on pages rather than bytes, implements custom caching strategies, and often operates on raw disk partitions. POSIX AIO doesn't dictate buffering rules though, so one can't implement it on top of Linux AIO. Linux AIO provides asynchronous equivalents of **fsync()/fdatasync()**, and vectored read/write operations. Other syscalls, like **fcntl()**, are not part of AIO.

Linux AIO has two notification channels. When an operation completes, the kernel can send your process a signal, as in POSIX AIO, or deliver the event via **eventfd**. The latter is a Linux-specific mechanism, which plays well with **epoll()** and friends. This means it's easy to plug **eventfd** into an existing event loop. As you may guess, this makes **eventfd** a preferred way to get Linux AIO notifications in your code.

Let's rewrite our example using native Linux AIO. At the beginning, we create the "context" with **io_setup()**:

```

#include <libaio.h>
#define NR_EVENTS 32
int main(int argc, char **argv)
{
    io_context_t ctx = { 0 };
}

```

When everything is a file

If you want something to integrate nicely with event loops, it should be available as a pollable file descriptor. That's why Linux wraps many traditional concepts, like signals, into file descriptors.

We briefly touched this topic in LV016. Now, let's focus on two mechanisms: **eventfd** and **signalfd**. Both can be used together with Linux AIO (albeit **eventfd** is preferred); **signalfd** may also be helpful in POSIX AIO. Note these mechanisms are Linux-specific, and you can't use them in portable code.

Eventfd is a generic event notification tool. It maintains a counter, which **write()** increases or sets, depending on its mode. **read()** either decrements the counter or resets it, and blocks if the counter is zero. **epoll()** and friends report the **eventfd** descriptor readable if it has an event pending. For more details, see **eventfd(2)**.

Signalfd works in a similar fashion. You create it with **signalfd()** and specify which signals you want to receive through the descriptor. Usually, you also block "normal" delivery of these signals with **sigprocmask()**. The descriptor becomes readable if there are any of these signals pending, and **read()** yields one or more **struct signalfd_siginfo** instances. The latter is mostly analogous to **siginfo_t** a "normal" signal handler receives. Again, refer to **signalfd(2)** for details.


```
char *buf = NULL;
int fd;
io_setup(NR_EVENTS, &ctx);
fd = open(argv[1], O_RDONLY | O_DIRECT);
posix_memalign((void **)&buf, 512, LARGE_CHUNK)
```

NR_EVENTS is how many concurrent AIO operations we want the context to support. **libaio** uses **io_context_t** as the context type, but native syscalls use **aio_context_t**. You must zero-fill either one, or **io_setup()** will complain. We also open the file and allocate a properly aligned memory buffer with **posix_memalign()**.

Next step is to prepare the request. Linux AIO represents requests with **struct iocb**. Conceptually, it's the same as **struct aiocb**, but **libaio** provides helper functions to fill it:

```
io_prep_pread(&request, fd, buf, LARGE_CHUNK, 0);
io_set_callback(&request, handle_pread);
res_fd = eventfd(0, EFD_NONBLOCK);
io_set_eventfd(&request, res_fd);
io_submit(ctx, 1, &request_ptr);
```

We also create a non-blocking **eventfd** and associate it with the request.

Finally, **io_submit()** is called to submit the request. The second argument is the number of requests to submit, so one can enqueue them in batches. A tricky part is that **io_submit()** can also block for various reasons, like allocating something in the kernel. In this toy program, it's not an issue, but in real code you'd probably call it inside the event loop. Another nuance is that you should never **fork()** with asynchronous operation in flight if you allocate buffers from the heap.

The only thing left is to start the event loop. With most of the **epoll()** machinery omitted (LV016 has all details), it may look like this:

```
while (!done) {
    n_ready = epoll_wait(epoll_fd, &epoll_event, 1, LOOP_TIMEOUT_MSEC);
    if (n_ready > 0) {
        if (epoll_event.events & EPOLLIN) {
            read(res_fd, eventfd_buf, sizeof(eventfd_buf));
            n_events = io_getevents(ctx, 0, 1, &io_event, NULL);
```

```
if (n_events > 0) {
    cb = (io_callback_t)event->data;
    iocb = event->obj;
    cb(ctx, iocb, event->res, event->res2);
}
}
```

We wait for **eventfd** to trigger, and resume pending events as soon as it can to let it trigger again. Then, **io_getevents()** is called. It can block if no I/O events are pending in the context, but as **eventfd** has fired, we know there is something in. **io_getevents()** accepts the context, and a minimum and maximum number of events to grab. It returns the number of events stored in the array that the **events** argument points to. If it's non-zero, we execute the callback. **event->data** comes from **request.data**, which **io_set_callback()** filled with a pointer to the **handle_pread()** function. As before, **handle_pread()** checks for errors and prints the number of bytes read. Refer to the sources for details. At the end of the program, we call **io_destroy()** to deallocate resources the context owns.

Several open-source projects implement Linux AIO support in their software. **MySQL** does this via **libaio** for the **InnoDB** storage engine. **Rspamd** (www.rspamd.com) builds on the raw Linux AIO syscall interface. It doesn't actually use AIO anywhere in the code now, but has the test suite providing nice usage examples. Database heavyweights like **Oracle** support Linux AIO as well, albeit they won't let you learn from them.

iotop gives quick and accurate answers to many questions regarding disk I/O in your system.

Command of the month: **iotop**

You probably know the **top** utility, which builds nice stats on CPU usage. **iotop** does the same, but for disk I/O. You must run **iotop** as root for security reasons. The main screen shows the overall system stats and the list of processes. **TOTAL** figures display the amount of data transferred to a kernel block device subsystem. **ACTUAL** measures the exchange with physical storage, and it may not be equal to **TOTAL** because the kernel caches data and reorders I/O requests for optimum performance.

Below it is per-process statistics. To toggle processes not doing any I/O now, press **O**, or run the tool as **iotop --only**. The **IO** column gives a clue on

how much time the process or thread slept waiting for blocking I/O to complete. **SWAPIN** is the same, but for swap. Use arrow keys to change the sorting column (the current one is marked with **>**), and **R** key to reverse sorting order.

By default, **iotop** displays an instant snapshot of the system's I/O. To show accumulated values instead, run it as **iotop -a** or press **A**. This is useful to see which task consumed most I/O bandwidth in the long run. It is also possible to run **iotop** in batch (non-interactive) mode and save its output to log files (LV025). In interactive mode, press **I** to assign a thread different I/O priority. **ionice(1)** describes the options.

/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!

So, Microsoft has decided to port one of its flagship products, *SQL Server*, to Linux. By 2017, you could all be querying MS's finest for your *WordPress* blogs if you so wish (and want to stump up the cash). We are expected to believe this is part of some "Redmond Spring" where friendly competition is embraced and not compared to deadly diseases.

It is perhaps just as likely though that the not-so-evil empire has realised that for *SQL Server* to compete in future, it really has to run on Linux. All of its main competitors do. And where once the argument could be made that people choose services, not operating systems, that doesn't hold true in the cloud to the same extent. And the cloud is where everything is.

Linux basically beats everything in the cloud (<http://goo.gl/FFHpm1>), and Linux images are among the most popular even on Microsoft's own cloud platform, Azure. So the decision to port *SQL Server* to what is ultimately the only growing market segment in which it can't compete at the moment is a bit of a no brainer. And the competition will do everyone good – *SQL Server* is fast and robust (at least on Windows), so no doubt Oracle is examining its strategy and code also. It's such a good idea, and it's interesting to think of what might follow (Active Directory anyone?). But still, there is a tingle of that old Ice Fear – the hurrahs over this announcement had barely died down when MS also revealed it had received its latest dollop of what it likes to euphemistically call "IP licensing fees" from another Android-based tablet manufacturer. Microsoft certainly loves Linux, all the way to the bank.



MY LINUX SETUP TOM ORALS

Regular reader and full time Linux user who lives by a river!

Q What version of Linux are you currently using?

A I'm using Arch Linux. I know it's become a little notorious for name dropping, but I love the package availability and git-based AURs.

Q And what desktop are you using at the moment?

A KDE – although the more I use it the less of KDE I'm using. I mostly use it to tile windows.

Q What was the first Linux setup you ever used?

A It was Mandrake back in the late 90s. I've used KDE ever since.

Q What Free Software/open source can't you live without?

A The kernel? Other than that, *Bash*, perhaps.

Q What do other people love but you can't get on with?

A I really don't like Unity. There's nothing visually appealing about the icons or the palette. It looks and feels too clunky for me, like a kiddie version of Linux. ☹

LINUXVOICE

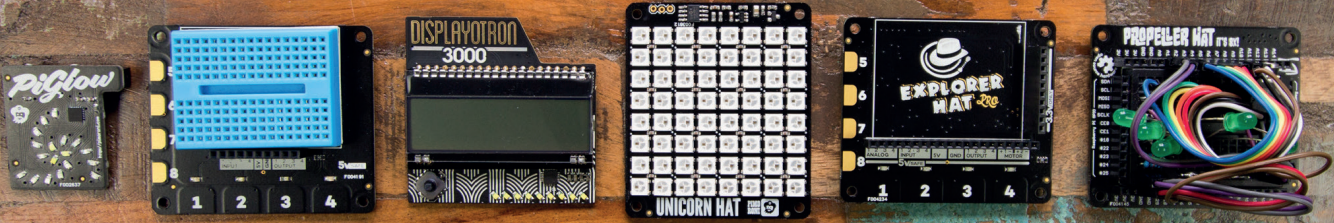
This is what we've done in the last 24 issues.
Subscribe to the next 12 from just **£38**.



Every subscription includes access to every PDF, ePub and audio edition we've ever published.

shop.linuxvoice.com

make something! :D



PIMORONI

<http://pimoroni.com>

<http://pimoroni.de>

Join the pirates!
<http://pimoroni.com/invest>

#CapitalAtRisk #InvestAware

For all things Pi <3



We also need to tell you: Investments of this nature carry risks as well as potential rewards. Please #InvestAware as your capital is at risk.

Approved as a financial promotion by Crowdcube Ventures Ltd, which is authorised & regulated by the Financial Conduct Authority (No. 572026).